

Gross-Pitaevskii-FDM

0.1

Generated by Doxygen 1.8.17

1 Hierarchical Index	1
1.1 Class Hierarchy	1
2 Class Index	3
2.1 Class List	3
3 File Index	5
3.1 File List	5
4 Class Documentation	7
4.1 BaseDomain Class Reference	7
4.1.1 Detailed Description	10
4.1.2 Constructor & Destructor Documentation	10
4.1.2.1 BaseDomain() [1/2]	11
4.1.2.2 BaseDomain() [2/2]	11
4.1.2.3 ~BaseDomain()	11
4.1.3 Member Function Documentation	11
4.1.3.1 assign_initial_value()	11
4.1.3.2 assign_wave_function()	12
4.1.3.3 at()	13
4.1.3.4 generate_directory_name()	13
4.1.3.5 generate_single_txt_file()	14
4.1.3.6 get_current_time_index()	15
4.1.3.7 get_dt()	15
4.1.3.8 get_infinitesimal_distance1()	16
4.1.3.9 get_infinitesimal_distance2()	16
4.1.3.10 get_null_gridpt()	17
4.1.3.11 get_num_grid_1()	17
4.1.3.12 get_num_grid_2()	18
4.1.3.13 get_num_times()	18
4.1.3.14 get_path()	19
4.1.3.15 get_t_end()	19
4.1.3.16 get_t_start()	19
4.1.3.17 normalize()	20
4.1.3.18 print_directory_info()	20
4.1.3.19 reset()	21
4.1.3.20 time_at()	21
4.1.3.21 update_time()	21
4.1.4 Member Data Documentation	22
4.1.4.1 current_grid	22
4.1.4.2 current_time_index	22
4.1.4.3 dt	22
4.1.4.4 null_gridpt	22

4.1.4.5 num_grid_1	22
4.1.4.6 num_grid_2	23
4.1.4.7 num_times	23
4.1.4.8 old_grid	23
4.1.4.9 PATH	23
4.1.4.10 t_end	23
4.1.4.11 t_start	23
4.2 BasePotential Class Reference	24
4.2.1 Detailed Description	25
4.2.2 Constructor & Destructor Documentation	25
4.2.2.1 BasePotential()	25
4.2.3 Member Function Documentation	25
4.2.3.1 calculte_potential_in_grid()	25
4.2.3.2 get_name()	26
4.2.3.3 potential_function()	26
4.2.4 Member Data Documentation	27
4.2.4.1 name	27
4.3 BaseSolver Class Reference	28
4.3.1 Detailed Description	29
4.3.2 Constructor & Destructor Documentation	29
4.3.2.1 BaseSolver() [1/2]	30
4.3.2.2 BaseSolver() [2/2]	30
4.3.2.3 ~BaseSolver()	30
4.3.3 Member Function Documentation	30
4.3.3.1 temporal_equation()	30
4.3.4 Member Data Documentation	31
4.3.4.1 g	31
4.3.4.2 string_info	31
4.4 BaseSpatialGrid Class Reference	31
4.4.1 Detailed Description	33
4.4.2 Constructor & Destructor Documentation	33
4.4.2.1 BaseSpatialGrid() [1/2]	34
4.4.2.2 BaseSpatialGrid() [2/2]	34
4.4.2.3 ~BaseSpatialGrid()	34
4.4.3 Member Function Documentation	34
4.4.3.1 at()	35
4.4.3.2 get_infinitesimal_distance1()	35
4.4.3.3 get_infinitesimal_distance2()	36
4.4.3.4 normalize()	36
4.4.4 Member Data Documentation	37
4.4.4.1 infinitesimal_distance_1	37
4.4.4.2 infinitesimal_distance_2	37

4.4.4.3 num_grid_1	37
4.4.4.4 num_grid_2	37
4.4.4.5 spatial_data	38
4.5 BaseSweeper Class Reference	38
4.5.1 Detailed Description	41
4.5.2 Constructor & Destructor Documentation	41
4.5.2.1 BaseSweeper() [1/2]	41
4.5.2.2 BaseSweeper() [2/2]	41
4.5.2.3 ~BaseSweeper()	42
4.5.3 Member Function Documentation	42
4.5.3.1 generate_num_list()	42
4.5.3.2 get_end()	42
4.5.3.3 get_number_of_pts()	43
4.5.3.4 get_start()	43
4.5.3.5 get_value_from_idx()	43
4.5.3.6 set_CUDA_info()	44
4.5.3.7 set_MPI_info()	44
4.5.3.8 set_print_info()	45
4.5.3.9 set_save_data()	45
4.5.4 Member Data Documentation	45
4.5.4.1 CUDA_use	45
4.5.4.2 end	46
4.5.4.3 endpoint	46
4.5.4.4 gpu_limit	46
4.5.4.5 gpu_num	46
4.5.4.6 MPI_use	46
4.5.4.7 num	46
4.5.4.8 num_list	47
4.5.4.9 print_info	47
4.5.4.10 rank	47
4.5.4.11 save_data	47
4.5.4.12 size	47
4.5.4.13 start	47
4.6 CNRectSolver Class Reference	48
4.6.1 Detailed Description	50
4.6.2 Constructor & Destructor Documentation	50
4.6.2.1 CNRectSolver() [1/2]	50
4.6.2.2 CNRectSolver() [2/2]	50
4.6.3 Member Function Documentation	51
4.6.3.1 calculate_error()	51
4.6.3.2 get_potential_value()	52
4.6.3.3 initialize_guess_with_forward_euler()	53

4.6.3.4 solve() [1/2]	54
4.6.3.5 solve() [2/2]	56
4.6.3.6 solve_single_time() [1/2]	57
4.6.3.7 solve_single_time() [2/2]	58
4.6.3.8 temporal_equation()	60
4.6.3.9 update_guess()	61
4.6.4 Member Data Documentation	62
4.6.4.1 domain	62
4.6.4.2 g	63
4.6.4.3 guess	63
4.6.4.4 string_info	63
4.7 ConfigParser Class Reference	63
4.7.1 Detailed Description	64
4.7.2 Constructor & Destructor Documentation	64
4.7.2.1 ConfigParser()	64
4.7.2.2 ~ConfigParser()	64
4.7.3 Member Function Documentation	64
4.7.3.1 get_default()	65
4.7.3.2 parse()	66
4.8 DomainParameters Class Reference	72
4.8.1 Detailed Description	72
4.8.2 Member Data Documentation	72
4.8.2.1 domain_type	73
4.8.2.2 n_time	73
4.8.2.3 n_x	73
4.8.2.4 n_y	73
4.8.2.5 spatial_parameters	73
4.8.2.6 time_end	73
4.8.2.7 time_start	74
4.9 EquationParameters Class Reference	74
4.9.1 Detailed Description	74
4.9.2 Member Data Documentation	74
4.9.2.1 g	75
4.9.2.2 potential_parameters	75
4.9.2.3 potential_type	75
4.10 FERectSolver Class Reference	75
4.10.1 Detailed Description	78
4.10.2 Constructor & Destructor Documentation	78
4.10.2.1 FERectSolver() [1/2]	78
4.10.2.2 FERectSolver() [2/2]	78
4.10.3 Member Function Documentation	79
4.10.3.1 get_potential_value()	79

4.10.3.2 solve()	80
4.10.3.3 solve_single_time()	81
4.10.3.4 temporal_equation()	82
4.10.4 Member Data Documentation	83
4.10.4.1 domain	83
4.10.4.2 g	84
4.10.4.3 string_info	84
4.11 GridPoint Class Reference	84
4.11.1 Detailed Description	85
4.11.2 Constructor & Destructor Documentation	85
4.11.2.1 GridPoint() [1/2]	85
4.11.2.2 GridPoint() [2/2]	85
4.11.2.3 ~GridPoint()	85
4.11.3 Member Data Documentation	85
4.11.3.1 value	86
4.11.3.2 x	86
4.11.3.3 y	86
4.12 GSweeper Class Reference	86
4.12.1 Detailed Description	89
4.12.2 Constructor & Destructor Documentation	89
4.12.2.1 GSweeper() [1/2]	89
4.12.2.2 GSweeper() [2/2]	89
4.12.3 Member Function Documentation	90
4.12.3.1 generate_num_list()	90
4.12.3.2 get_end()	90
4.12.3.3 get_number_of_pts()	90
4.12.3.4 get_start()	91
4.12.3.5 get_value_from_idx()	91
4.12.3.6 run()	91
4.12.3.7 set_CUDA_info()	93
4.12.3.8 set_MPI_info()	94
4.12.3.9 set_print_info()	94
4.12.3.10 set_save_data()	95
4.12.4 Member Data Documentation	95
4.12.4.1 CUDA_use	95
4.12.4.2 end	96
4.12.4.3 endpoint	96
4.12.4.4 gpu_limit	96
4.12.4.5 gpu_num	96
4.12.4.6 MPI_use	96
4.12.4.7 num	96
4.12.4.8 num_list	97

4.12.4.9 print_info	97
4.12.4.10 rank	97
4.12.4.11 save_data	97
4.12.4.12 size	97
4.12.4.13 start	97
4.13 HarmonicPotential Class Reference	98
4.13.1 Detailed Description	99
4.13.2 Constructor & Destructor Documentation	100
4.13.2.1 HarmonicPotential()	100
4.13.3 Member Function Documentation	100
4.13.3.1 calculalte_potential_in_grid()	100
4.13.3.2 get_name()	101
4.13.3.3 potential_function()	102
4.13.4 Member Data Documentation	102
4.13.4.1 name	102
4.13.4.2 omega_x	102
4.13.4.3 omega_y	102
4.14 HPSweeper Class Reference	103
4.14.1 Detailed Description	105
4.14.2 Constructor & Destructor Documentation	105
4.14.2.1 HPSweeper() [1/2]	105
4.14.2.2 HPSweeper() [2/2]	105
4.14.3 Member Function Documentation	106
4.14.3.1 generate_num_list()	106
4.14.3.2 get_end()	106
4.14.3.3 get_number_of_pts()	106
4.14.3.4 get_start()	107
4.14.3.5 get_value_from_idx()	107
4.14.3.6 run()	107
4.14.3.7 set_CUDA_info()	109
4.14.3.8 set_MPI_info()	110
4.14.3.9 set_print_info()	110
4.14.3.10 set_save_data()	111
4.14.4 Member Data Documentation	111
4.14.4.1 CUDA_use	111
4.14.4.2 end	112
4.14.4.3 endpoint	112
4.14.4.4 gpu_limit	112
4.14.4.5 gpu_num	112
4.14.4.6 MPI_use	112
4.14.4.7 num	112
4.14.4.8 num_list	113

4.14.4.9 print_info	113
4.14.4.10 rank	113
4.14.4.11 save_data	113
4.14.4.12 size	113
4.14.4.13 start	113
4.15 InitialCondition Class Reference	114
4.15.1 Detailed Description	114
4.15.2 Constructor & Destructor Documentation	114
4.15.2.1 InitialCondition() [1/2]	114
4.15.2.2 InitialCondition() [2/2]	114
4.15.3 Member Function Documentation	115
4.15.3.1 assign_to_domain()	115
4.15.4 Member Data Documentation	116
4.15.4.1 initial_condition_function	116
4.16 InitialConditionParameters Class Reference	116
4.16.1 Detailed Description	117
4.16.2 Member Data Documentation	117
4.16.2.1 init_cond_parameters	117
4.16.2.2 init_cond_type	117
4.17 MainParameters Class Reference	117
4.17.1 Detailed Description	118
4.17.2 Member Data Documentation	118
4.17.2.1 calculation_type	118
4.17.2.2 float_parameters	118
4.17.2.3 int_parameters	118
4.18 Parameters Class Reference	119
4.18.1 Detailed Description	119
4.18.2 Member Data Documentation	119
4.18.2.1 config_name	119
4.18.2.2 domain_parameters	120
4.18.2.3 equation_parameters	120
4.18.2.4 init_cond_parameters	120
4.18.2.5 main_parameters	120
4.18.2.6 solver_parameters	120
4.19 RectangularDomain Class Reference	121
4.19.1 Detailed Description	124
4.19.2 Constructor & Destructor Documentation	124
4.19.2.1 RectangularDomain() [1/2]	124
4.19.2.2 RectangularDomain() [2/2]	124
4.19.2.3 ~RectangularDomain()	125
4.19.3 Member Function Documentation	125
4.19.3.1 assign_initial_value()	125

4.19.3.2 assign_wave_function()	126
4.19.3.3 at()	127
4.19.3.4 generate_directory_name()	127
4.19.3.5 generate_single_txt_file()	128
4.19.3.6 get_current_time_index()	129
4.19.3.7 get_dt()	130
4.19.3.8 get_infinitesimal_distance1()	130
4.19.3.9 get_infinitesimal_distance2()	131
4.19.3.10 get_null_gridpt()	131
4.19.3.11 get_num_grid_1()	132
4.19.3.12 get_num_grid_2()	132
4.19.3.13 get_num_times()	133
4.19.3.14 get_path()	133
4.19.3.15 get_t_end()	133
4.19.3.16 get_t_start()	133
4.19.3.17 get_x_end()	134
4.19.3.18 get_x_start()	134
4.19.3.19 get_y_end()	134
4.19.3.20 get_y_start()	134
4.19.3.21 normalize()	135
4.19.3.22 print_directory_info()	135
4.19.3.23 reset()	136
4.19.3.24 time_at()	136
4.19.3.25 update_time()	137
4.19.4 Member Data Documentation	137
4.19.4.1 current_grid	137
4.19.4.2 current_time_index	137
4.19.4.3 dt	138
4.19.4.4 null_gridpt	138
4.19.4.5 num_grid_1	138
4.19.4.6 num_grid_2	138
4.19.4.7 num_times	138
4.19.4.8 old_grid	138
4.19.4.9 PATH	139
4.19.4.10 potential_grid	139
4.19.4.11 t_end	139
4.19.4.12 t_start	139
4.19.4.13 x_end	139
4.19.4.14 x_start	139
4.19.4.15 y_end	140
4.19.4.16 y_start	140
4.20 RectangularSpatialGrid Class Reference	140

4.20.1 Detailed Description	143
4.20.2 Constructor & Destructor Documentation	143
4.20.2.1 RectangularSpatialGrid() [1/2]	143
4.20.2.2 RectangularSpatialGrid() [2/2]	143
4.20.2.3 ~RectangularSpatialGrid()	144
4.20.3 Member Function Documentation	144
4.20.3.1 at()	144
4.20.3.2 get_infinesimal_distance1()	145
4.20.3.3 get_infinesimal_distance2()	145
4.20.3.4 normalize()	145
4.20.4 Member Data Documentation	146
4.20.4.1 infinitesimal_distance_1	146
4.20.4.2 infinitesimal_distance_2	146
4.20.4.3 num_grid_1	147
4.20.4.4 num_grid_2	147
4.20.4.5 spatial_data	147
4.20.4.6 x_end	147
4.20.4.7 x_start	147
4.20.4.8 y_end	147
4.20.4.9 y_start	148
4.21 SolverParameters Class Reference	148
4.21.1 Detailed Description	148
4.21.2 Member Data Documentation	149
4.21.2.1 int_parameters	149
4.21.2.2 method	149
4.21.2.3 print_info	149
4.21.2.4 run_parallel	149
4.21.2.5 save_data	149
4.21.2.6 solver_parameters	149
5 File Documentation	151
5.1 /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/CMakeLists.txt File Reference	151
5.1.1 Function Documentation	151
5.1.1.1 cmake_minimum_required()	151
5.1.1.2 target_link_libraries()	152
5.2 /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/configuration/config_parser.cpp File Reference	152
5.2.1 Detailed Description	152
5.3 /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/configuration/config_parser.h File Reference	153
5.3.1 Detailed Description	154
5.4 /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/configuration/parameters.h File Reference	154
5.4.1 Detailed Description	155

5.5 /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/domain/base_domain.cpp File Reference	156
5.5.1 Detailed Description	156
5.6 /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/domain/base_domain.h File Reference	157
5.6.1 Detailed Description	158
5.7 /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/domain/rect_domain.cpp File Reference	158
5.7.1 Detailed Description	159
5.8 /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/domain/rect_domain.h File Reference	159
5.8.1 Detailed Description	160
5.9 /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/initial_condition/initial_condition.cpp File Reference	160
5.9.1 Detailed Description	161
5.10 /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/initial_condition/initial_condition.h File Reference	161
5.10.1 Detailed Description	162
5.11 /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/main.cpp File Reference	163
5.11.1 Function Documentation	163
5.11.1.1 main()	163
5.12 /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/potential/base_potential.cpp File Reference	167
5.12.1 Detailed Description	168
5.13 /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/potential/base_potential.h File Reference	168
5.13.1 Detailed Description	169
5.14 /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/potential/harmonic_potential.cpp File Reference	170
5.14.1 Detailed Description	170
5.15 /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/potential/harmonic_potential.h File Reference	171
5.15.1 Detailed Description	171
5.16 /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/solver/base_solver.cpp File Reference	172
5.16.1 Detailed Description	172
5.17 /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/solver/base_solver.h File Reference	173
5.17.1 Detailed Description	174
5.18 /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/solver/serial_solver/crank_nicolson/cn↔_rect_solver.cpp File Reference	174
5.18.1 Detailed Description	175
5.19 /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/solver/serial_solver/crank_nicolson/cn↔_rect_solver.h File Reference	175
5.19.1 Detailed Description	176
5.20 /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/solver/serial_solver/forward_euler/fe↔_rect_solver.cpp File Reference	176
5.20.1 Detailed Description	177
5.21 /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/solver/serial_solver/forward_euler/fe↔_rect_solver.h File Reference	177
5.21.1 Detailed Description	178

5.22 /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/sweeper/base_sweeper.cpp File Reference	179
5.23 /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/sweeper/base_sweeper.h File Reference	180
5.24 /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/sweeper/g_sweeper.cpp File Reference	180
5.25 /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/sweeper/g_sweeper.h File Reference	181
5.26 /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/sweeper/harmonic_p_sweeper.cpp File Reference	182
5.27 /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/sweeper/harmonic_p_sweeper.h File Reference	183
5.28 /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/utils.cpp File Reference	184
5.28.1 Function Documentation	184
5.28.1.1 is_close()	184
5.29 /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/utils.h File Reference	185
5.29.1 Function Documentation	185
5.29.1.1 is_close()	186
Index	187

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

BaseDomain	7
RectangularDomain	121
BasePotential	24
HarmonicPotential	98
BaseSolver	28
FERectSolver	75
CNRectSolver	48
BaseSpatialGrid	31
RectangularSpatialGrid	140
BaseSweeper	38
GSweeper	86
HPSweeper	103
ConfigParser	63
DomainParameters	72
EquationParameters	74
GridPoint	84
InitialCondition	114
InitialConditionParameters	116
MainParameters	117
Parameters	119
SolverParameters	148

Chapter 2

Class Index

2.1 Class List

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

BaseDomain	
Base domain class containing multiple time steps and export methods	7
BasePotential	24
BaseSolver	
Base Solver for Gross Piteavskill finite difference solver	28
BaseSpatialGrid	
Base Spatial Grid class for single time step	31
BaseSweeper	38
CNRectSolver	48
ConfigParser	
Configuration parser	63
DomainParameters	
Domain branch parameters containing information about domain	72
EquationParameters	
Equation branch parameters containing information about equation	74
FERectSolver	
Forward Euler Serial Solver	75
GridPoint	
Grid point class (single point)	84
GSweeper	86
HarmonicPotential	98
HPSweeper	103
InitialCondition	
Initial condition class	114
InitialConditionParameters	
Initial condition branch parameters containing information about initial condition	116
MainParameters	
Main branch parameters containing information about overall calculation	117
Parameters	
Parameters containing configuration name and all other branched parameters	119
RectangularDomain	
Rectangular domain containing multiple timesteps	121
RectangularSpatialGrid	
Rectangular Spatial Grid class for single time step	140
SolverParameters	
Solver branch parameters containing information about solver	148

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

/home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/main.cpp	163
/home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/utlis.cpp	184
/home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/utlis.h	185
/home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/configuration/config_parser.cpp Configuration Parser class implementation	152
/home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/configuration/config_parser.h Configuration Parser Class header	153
/home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/configuration/parameters.h Header for the Parameter classes	154
/home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/domain/base_domain.cpp Implementation of methods in the Base Domain class	156
/home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/domain/base_domain.h Header for Base domain class	157
/home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/domain/rect_domain.cpp Implementation of methods in the Rectangular Domain class	158
/home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/domain/rect_domain.h Header for Rectangular domain class	159
/home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/initial_condition/initial_condition.cpp Implementation of initial condition class	160
/home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/initial_condition/initial_condition.h Header for the initial condition class	161
/home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/potential/base_potential.cpp Implementation of the methods in potential class	167
/home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/potential/base_potential.h Header for the base potential class. Heritate this class for create custom potential shape . . .	168
/home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/potential/harmonic_potential.cpp Implementation of harmonic potential methods	170
/home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/potential/harmonic_potential.h Header file for harmoni potential ($V = 0.5 * (\omega_x^2 * x^2 + \omega_y^2 * y^2)$)	171
/home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/solver/base_solver.cpp Header file for base solver	172
/home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/solver/base_solver.h Implementation file for base solver	173
/home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/solver/serial_solver/crank_nicolson/cn_rect_solver.cpp Implementation file for serial crank nicolson solver	174

/home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/solver/serial_solver/crank_nicolson/cn_rect_solver.h	
Header file for serial crank nicolson solver	175
/home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/solver/serial_solver/forward_euler/fe_rect_solver.cpp	
Implementation file for serial forward euler solver methods	176
/home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/solver/serial_solver/forward_euler/fe_rect_solver.h	
Header file for serial forward euler solver	177
/home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/sweeper/base_sweeper.cpp	179
/home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/sweeper/base_sweeper.h	180
/home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/sweeper/g_sweeper.cpp	180
/home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/sweeper/g_sweeper.h	181
/home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/sweeper/harmonic_p_sweeper.cpp	182
/home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/sweeper/harmonic_p_sweeper.h	183

Chapter 4

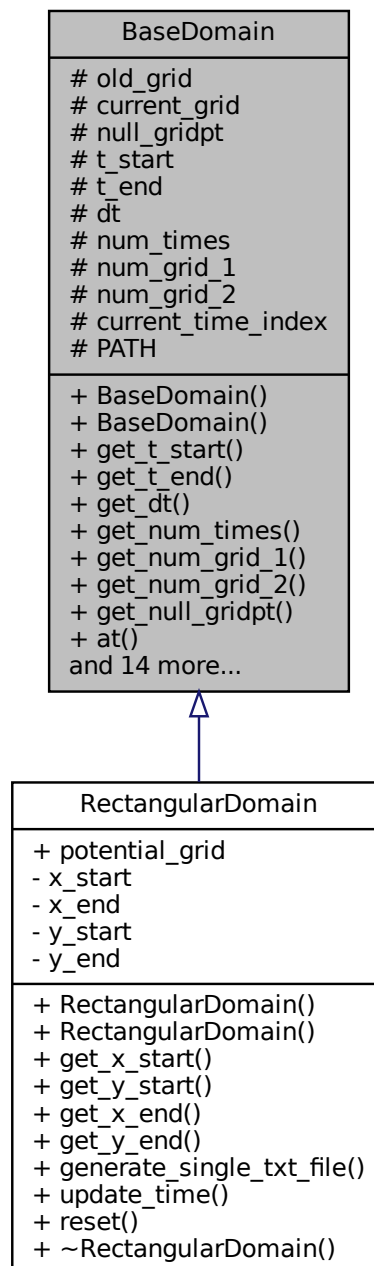
Class Documentation

4.1 BaseDomain Class Reference

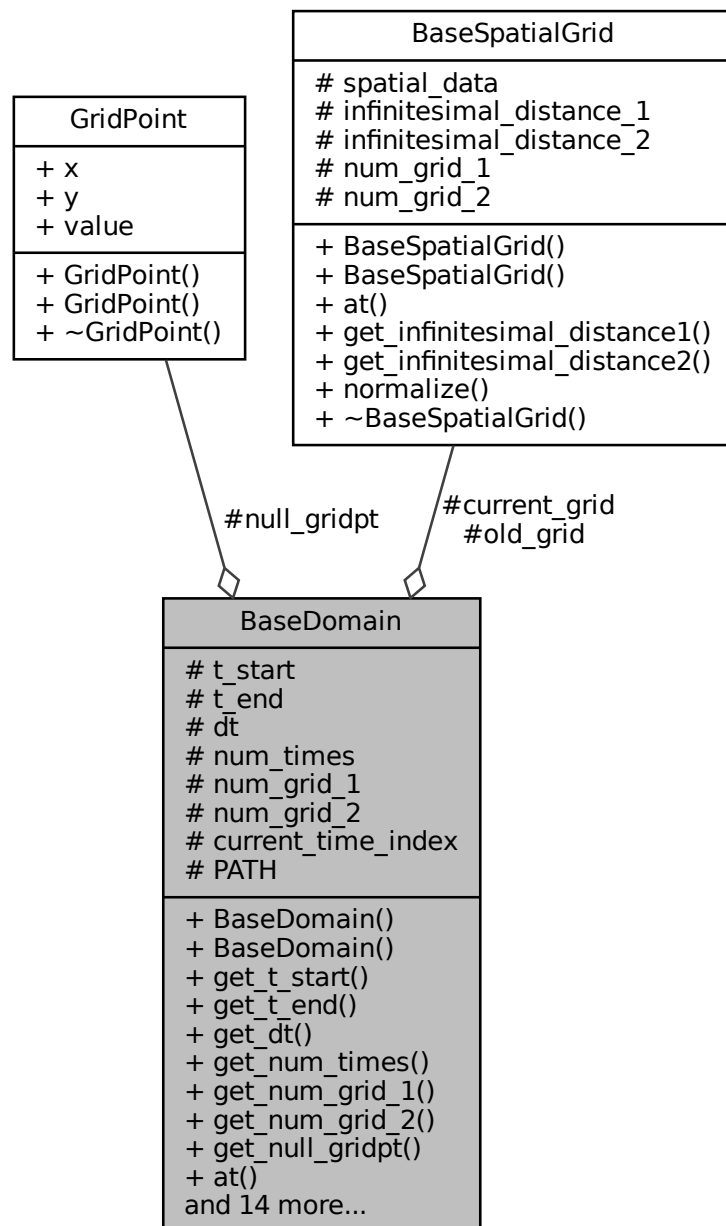
Base domain class containing multiple time steps and export methods.

```
#include <base_domain.h>
```

Inheritance diagram for BaseDomain:



Collaboration diagram for BaseDomain:



Public Member Functions

- [BaseDomain \(\)](#)
- [BaseDomain \(int num_grid_1, int num_grid_2, float t_start, float t_end, int num_times\)](#)
- float [get_t_start \(\)](#)
- float [get_t_end \(\)](#)
- float [get_dt \(\)](#)
- int [get_num_times \(\)](#)

- int [get_num_grid_1](#) ()
- int [get_num_grid_2](#) ()
- [GridPoint](#) * [get_null_gridpt](#) ()
- [GridPoint](#) * [at](#) (int index_1, int index_2, int time_index)
- void [assign_initial_value](#) (int index_1, int index_2, std::complex< float > value)
Assign initial value.
- void [assign_wave_function](#) (int index_1, int index_2, int time_index, std::complex< float > value)
- float [time_at](#) (int time_index)
- float [get_infinitesimal_distance1](#) ()
- float [get_infinitesimal_distance2](#) ()
- void [generate_directory_name](#) (std::string info, bool print_info=true)
create directory and return directory name with "/" directory name : "./results/%d-%m-%Y %H-%M-%S_"+info for exmaple, "./results/"
- void [generate_single_txt_file](#) (std::string filename, bool cuda_mode=false)
export results of probability($|\psi|^2$) as txt it generates (num_times) txt files each txt file contains $|\psi|^2$ data
- void [normalize](#) (int time_index)
- void [print_directory_info](#) ()
- int [get_current_time_index](#) ()
- [~BaseDomain](#) ()
- std::string [get_path](#) ()
- void [update_time](#) (bool cuda_mode=false)
- virtual void [reset](#) ()

Protected Attributes

- [BaseSpatialGrid](#) * [old_grid](#)
- [BaseSpatialGrid](#) * [current_grid](#)
- [GridPoint](#) * [null_gridpt](#)
- float [t_start](#)
- float [t_end](#)
- float [dt](#)
- int [num_times](#)
- int [num_grid_1](#)
- int [num_grid_2](#)
- int [current_time_index](#) = 0
- std::string [PATH](#)

4.1.1 Detailed Description

Base domain class containing multiple time steps and export methods.

Definition at line 67 of file base_domain.h.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 BaseDomain() [1/2]

```
BaseDomain::BaseDomain ( )
```

4.1.2.2 BaseDomain() [2/2]

```
BaseDomain::BaseDomain (
    int num_grid_1,
    int num_grid_2,
    float t_start,
    float t_end,
    int num_times )
```

Definition at line 81 of file base_domain.cpp.

```
87 {
88     this->t_start = t_start;
89     this->t_end = t_end;
90     this->num_times = num_times;
91     // dt: interval of time
92     this->dt = (t_end - t_start) / (num_times - 1);
93     this->num_grid_1 = num_grid_1;
94     this->num_grid_2 = num_grid_2;
95     this->old_grid = new BaseSpatialGrid(num_grid_1, num_grid_2);
96     this->current_grid = new BaseSpatialGrid(num_grid_1, num_grid_2);
97     this->null_gridpt = new GridPoint(0, 0, std::complex<float>{0});
98 }
```

4.1.2.3 ~BaseDomain()

```
BaseDomain::~~BaseDomain ( )
```

Definition at line 100 of file base_domain.cpp.

```
101 {
102     delete (this->old_grid);
103     delete (this->current_grid);
104 }
```

4.1.3 Member Function Documentation**4.1.3.1 assign_initial_value()**

```
void BaseDomain::assign_initial_value (
    int index_1,
    int index_2,
    std::complex< float > value )
```

Assign initial value.

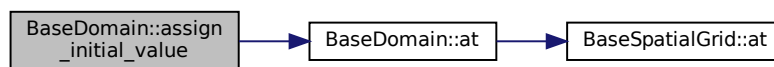
Parameters

<i>index_1</i>	$x = x_start + index_1 * dx$
<i>index_2</i>	$y = y_start + index_2 * dy$
<i>value</i>	initial value at x, y

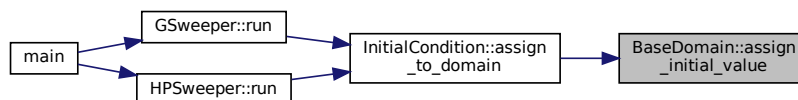
Definition at line 188 of file base_domain.cpp.

```
189 {
190     this->at(index_1, index_2, 0)->value = value;
191 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



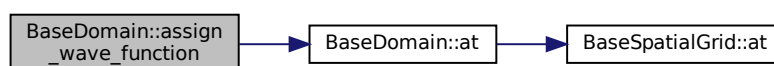
4.1.3.2 assign_wave_function()

```
void BaseDomain::assign_wave_function (
    int index_1,
    int index_2,
    int time_index,
    std::complex< float > value )
```

Definition at line 192 of file base_domain.cpp.

```
193 {
194     this->at(index_1, index_2, time_index)->value = value;
195 }
```

Here is the call graph for this function:



4.1.3.3 at()

```
GridPoint * BaseDomain::at (
    int index_1,
    int index_2,
    int time_index )
```

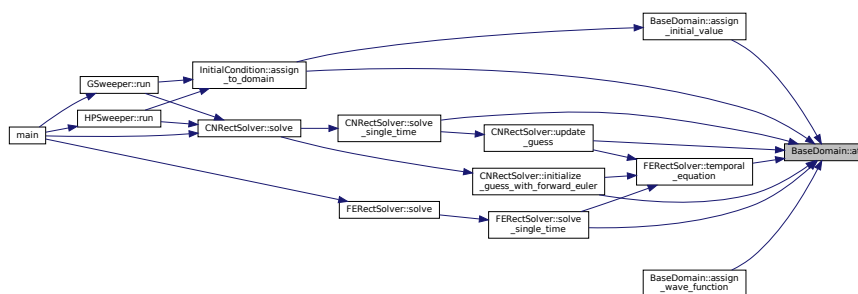
Definition at line 153 of file base_domain.cpp.

```
154 {
155     if (time_index == this->current_time_index)
156     {
157         return this->current_grid->at(index_1, index_2);
158     }
159     else if (time_index == this->current_time_index - 1)
160     {
161         return this->old_grid->at(index_1, index_2);
162     }
163     else
164     {
165         std::cerr << "error in base domin at function" << std::endl;
166         return this->current_grid->at(index_1, index_2);
167     }
168 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.1.3.4 generate_directory_name()

```
void BaseDomain::generate_directory_name (
    std::string info,
    bool print_info = true )
```

create directory and return directory name with "/" directory name : `"/results/%d-%m-%Y %H-%M-%S_"`+info for exmaple, `"/results/"`

Parameters

<i>info</i>	information about domain type and solver type
-------------	---

Returns

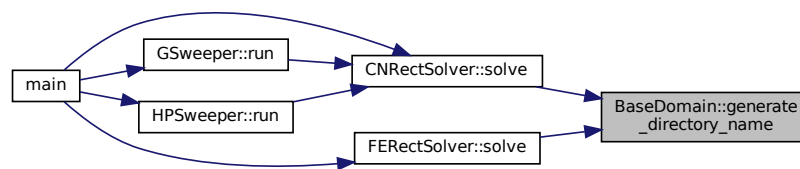
std::string directory name with "/"

Definition at line 203 of file base_domain.cpp.

```

204 {
205     auto t = std::time(nullptr);
206     auto tm = *std::localtime(&t);
207
208     std::ostringstream oss;
209     oss << std::put_time(&tm, "%Y-%m-%d-%H-%M-%S");
210     auto str = oss.str();
211     std::string directory_name = "../results/" + str + "_" + info;
212     bool created = fs::create_directory(directory_name.c_str());
213     if (print_info && created)
214     {
215         std::cout << "Created directory " << directory_name << std::endl;
216     }
217     // else if(print_info )
218     //TODO
219     else if (!created)
220     {
221         std::cout << "Creating directory failed" << std::endl;
222     }
223
224     this->PATH = directory_name + "/";
225 }
```

Here is the caller graph for this function:



4.1.3.5 generate_single_txt_file()

```

void BaseDomain::generate_single_txt_file (
    std::string filename,
    bool cuda_mode = false )
```

export results of probability($|\psi|^2$) as txt it generates (num_times) txt files each txt file contains $|\psi|^2$ data

Parameters

<i>grid</i>	grid at t
<i>filename</i>	file name to store data

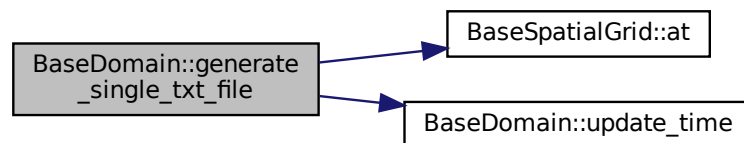
Definition at line 235 of file base_domain.cpp.

```

236 {
237     if (cuda_mode){
238         this->update_time(cuda_mode);
239     }
240     else{
241         std::ofstream outfile(this->PATH + filename + ".txt");
242         outfile << "x, y, real, imag, magn, phase " << std::endl;
243         for (auto i = 0; i < num_grid_1; ++i)
244         {
245             for (auto j = 0; j < num_grid_2; ++j)
246             {
247                 float magnitude = std::abs(this->current_grid->at(i, j)->value);
248                 float phase = std::arg(this->current_grid->at(i, j)->value);
249                 outfile << this->current_grid->at(i, j)->x << ", " << this->current_grid->at(i, j)->y << ",
";
250                 outfile << this->current_grid->at(i, j)->value.real() << ", " << this->current_grid->at(i,
j)->value.imag() << ", ";
251                 outfile << magnitude << ", " << phase;
252                 outfile << std::endl;
253             }
254         }
255         outfile.close();
256         // After saving data, update domain
257         this->update_time();
258     }
259 };

```

Here is the call graph for this function:



4.1.3.6 get_current_time_index()

```
int BaseDomain::get_current_time_index ( )
```

Definition at line 170 of file base_domain.cpp.

```

171 {
172     return this->current_time_index;
173 }

```

4.1.3.7 get_dt()

```
float BaseDomain::get_dt ( )
```

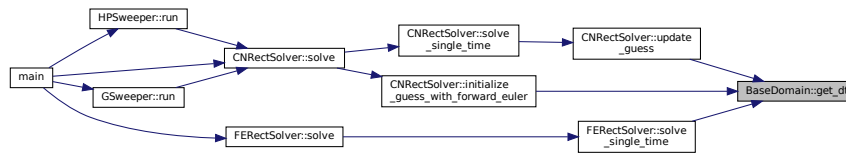
Definition at line 125 of file base_domain.cpp.

```

126 {
127     return this->dt;
128 }

```

Here is the caller graph for this function:



4.1.3.8 get_infinitesimal_distance1()

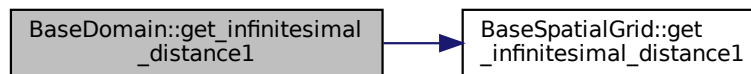
```
float BaseDomain::get_infinitesimal_distance1 ( )
```

Definition at line 143 of file base_domain.cpp.

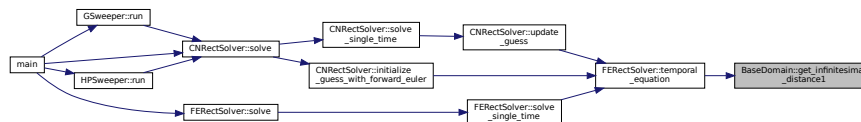
```

144 {
145     return this->old_grid->get_infinitesimal_distance1();
146 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.1.3.9 get_infinitesimal_distance2()

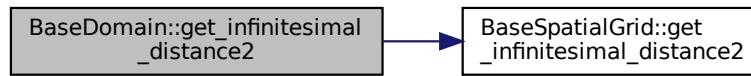
```
float BaseDomain::get_infinitesimal_distance2 ( )
```

Definition at line 147 of file base_domain.cpp.

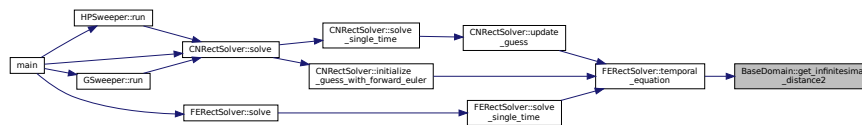
```

148 {
149     return this->old_grid->get_infinitesimal_distance2();
150 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.1.3.10 get_null_gridpt()

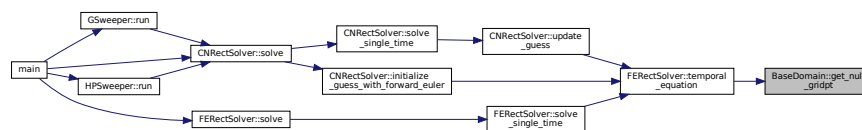
```
GridPoint * BaseDomain::get_null_gridpt ( )
```

Definition at line 176 of file base_domain.cpp.

```

177 {
178     return this->null_gridpt;
179 }
```

Here is the caller graph for this function:



4.1.3.11 get_num_grid_1()

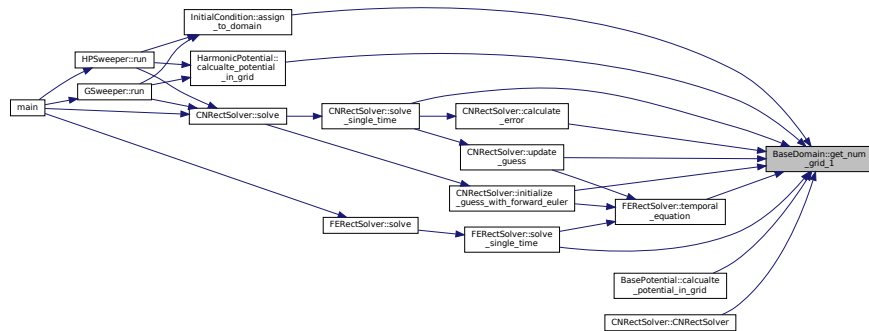
```
int BaseDomain::get_num_grid_1 ( )
```

Definition at line 134 of file base_domain.cpp.

```

135 {
136     return this->num_grid_1;
137 }
```

Here is the caller graph for this function:



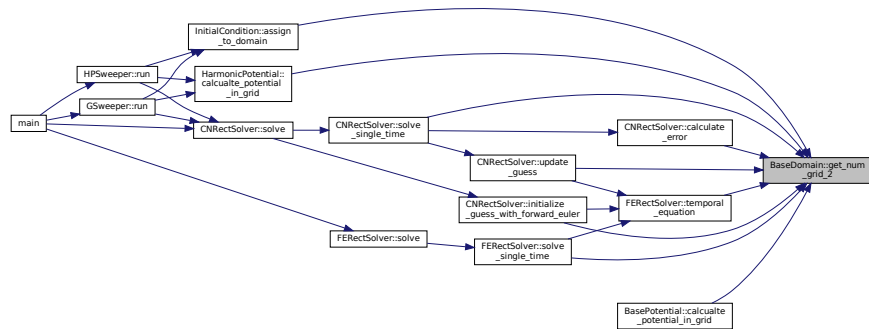
4.1.3.12 get_num_grid_2()

```
int BaseDomain::get_num_grid_2 ( )
```

Definition at line 138 of file base_domain.cpp.

```
139 {
140     return this->num_grid_2;
141 }
```

Here is the caller graph for this function:



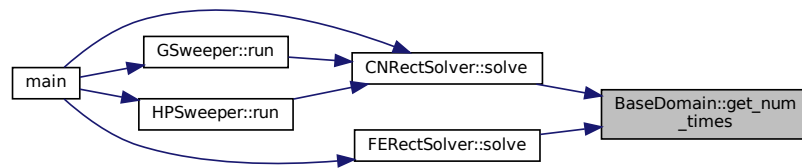
4.1.3.13 get_num_times()

```
int BaseDomain::get_num_times ( )
```

Definition at line 129 of file base_domain.cpp.

```
130 {
131     return this->num_times;
132 }
```


Here is the caller graph for this function:



4.1.3.14 get_path()

```
std::string BaseDomain::get_path ( )
```

Definition at line 106 of file `base_domain.cpp`.

```
107 {
108     return this->PATH;
109 }
```

4.1.3.15 get_t_end()

```
float BaseDomain::get_t_end ( )
```

Definition at line 121 of file `base_domain.cpp`.

```
122 {
123     return this->t_end;
124 }
```

4.1.3.16 get_t_start()

```
float BaseDomain::get_t_start ( )
```

Definition at line 116 of file `base_domain.cpp`.

```
117 {
118     return this->t_start;
119 }
```

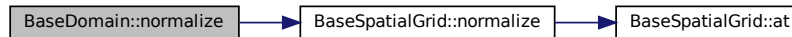
4.1.3.17 normalize()

```
void BaseDomain::normalize (
    int time_index )
```

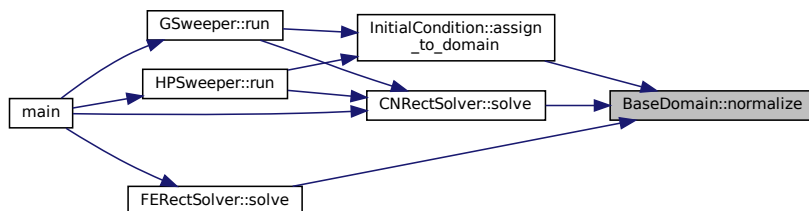
Definition at line 111 of file base_domain.cpp.

```
112 {
113     this->current_grid->normalize();
114 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



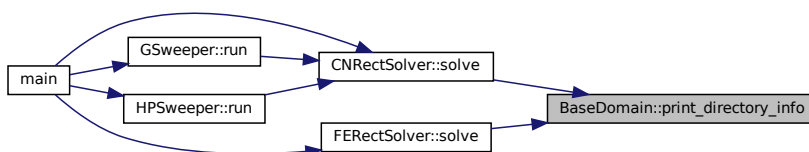
4.1.3.18 print_directory_info()

```
void BaseDomain::print_directory_info ( )
```

Definition at line 261 of file base_domain.cpp.

```
262 {
263     std::cout << this->num_times;
264     std::cout << " text files are generated in \n";
265     std::cout << this->PATH << std::endl;
266 }
```

Here is the caller graph for this function:



4.1.3.19 reset()

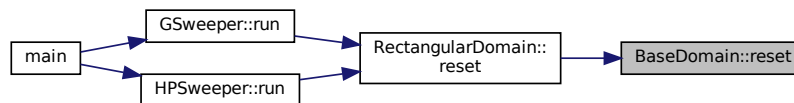
```
void BaseDomain::reset ( ) [virtual]
```

Reimplemented in [RectangularDomain](#).

Definition at line 282 of file `base_domain.cpp`.

```
283 {
284     this->current_time_index = 0;
285     delete this->old_grid;
286     delete this->current_grid;
287
288     this->current_grid = new BaseSpatialGrid(num_grid_1, num_grid_2);
289     this->old_grid = new BaseSpatialGrid(num_grid_1, num_grid_2);
290 }
```

Here is the caller graph for this function:



4.1.3.20 time_at()

```
float BaseDomain::time_at (
    int time_index )
```

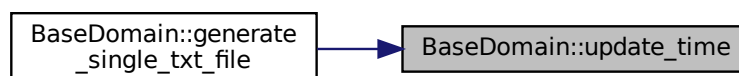
4.1.3.21 update_time()

```
void BaseDomain::update_time (
    bool cuda_mode = false )
```

Definition at line 268 of file `base_domain.cpp`.

```
269 {
270     if (cuda_mode){
271         this->current_time_index += 1;
272     }
273     else{
274
275         this->current_time_index += 1;
276         delete (this->old_grid);
277         this->old_grid = this->current_grid;
278         this->current_grid = new BaseSpatialGrid(num_grid_1, num_grid_2);
279     }
280 }
```

Here is the caller graph for this function:



4.1.4 Member Data Documentation

4.1.4.1 current_grid

```
BaseSpatialGrid* BaseDomain::current_grid [protected]
```

Definition at line 98 of file base_domain.h.

4.1.4.2 current_time_index

```
int BaseDomain::current_time_index = 0 [protected]
```

Definition at line 102 of file base_domain.h.

4.1.4.3 dt

```
float BaseDomain::dt [protected]
```

Definition at line 100 of file base_domain.h.

4.1.4.4 null_gridpt

```
GridPoint* BaseDomain::null_gridpt [protected]
```

Definition at line 99 of file base_domain.h.

4.1.4.5 num_grid_1

```
int BaseDomain::num_grid_1 [protected]
```

Definition at line 101 of file base_domain.h.

4.1.4.6 num_grid_2

```
int BaseDomain::num_grid_2 [protected]
```

Definition at line 101 of file base_domain.h.

4.1.4.7 num_times

```
int BaseDomain::num_times [protected]
```

Definition at line 101 of file base_domain.h.

4.1.4.8 old_grid

```
BaseSpatialGrid* BaseDomain::old_grid [protected]
```

Definition at line 97 of file base_domain.h.

4.1.4.9 PATH

```
std::string BaseDomain::PATH [protected]
```

Definition at line 103 of file base_domain.h.

4.1.4.10 t_end

```
float BaseDomain::t_end [protected]
```

Definition at line 100 of file base_domain.h.

4.1.4.11 t_start

```
float BaseDomain::t_start [protected]
```

Definition at line 100 of file base_domain.h.

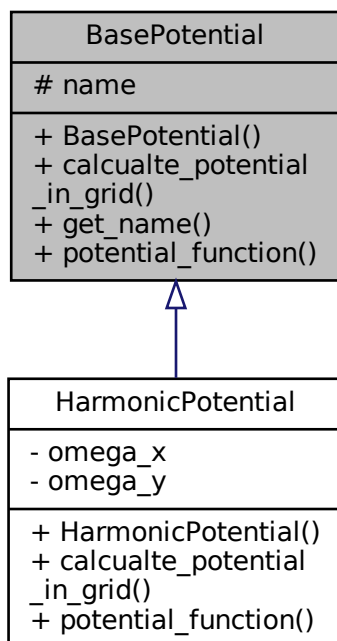
The documentation for this class was generated from the following files:

- [/home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/domain/base_domain.h](#)
- [/home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/domain/base_domain.cpp](#)

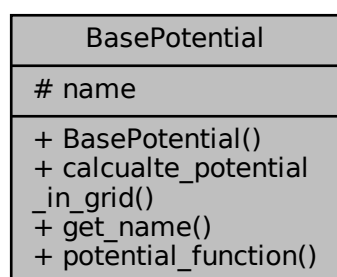
4.2 BasePotential Class Reference

```
#include <base_potential.h>
```

Inheritance diagram for BasePotential:



Collaboration diagram for BasePotential:



Public Member Functions

- [BasePotential](#) ()=default
- void [calcuale_potential_in_grid](#) ([RectangularDomain](#) *domain)
Calculate potential values in each grid based on domain point coordinate and potential function.
- std::string [get_name](#) ()
Getter for name class variable.
- float [potential_function](#) (float, float)
Default potential function for constant zero potential.

Protected Attributes

- std::string [name](#)

4.2.1 Detailed Description

Definition at line 18 of file base_potential.h.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 BasePotential()

```
BasePotential::BasePotential ( ) [default]
```

4.2.3 Member Function Documentation

4.2.3.1 calcuale_potential_in_grid()

```
void BasePotential::calcuale_potential_in_grid (
    RectangularDomain * domain )
```

Calculate potential values in each grid based on domain point coordinate and potential function.

Parameters

<i>domain</i>	
---------------	--

Definition at line 18 of file base_potential.cpp.

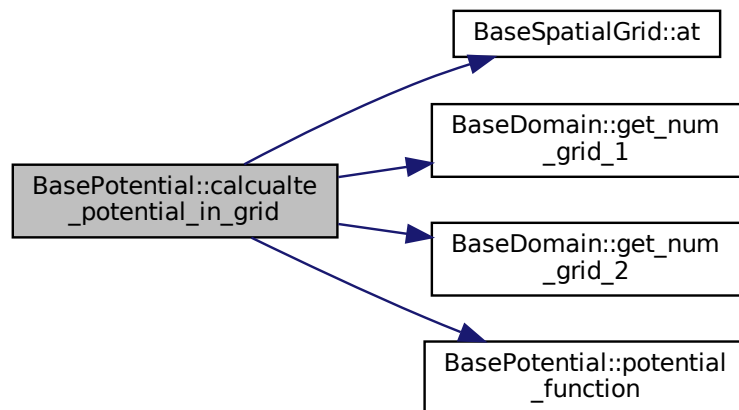
```
19 {
20     int num_grid_1 = domain->get_num_grid_1();
21     int num_grid_2 = domain->get_num_grid_2();
```

```

22     for (auto i = 0; i < num_grid_1; ++i)
23     {
24         for (auto j = 0; j < num_grid_2; ++j)
25         {
26             auto point = domain->potential_grid->at(i, j);
27
28             //Assign potential value in potential grid
29             point->value = std::complex<float>{this->potential_function(point->x, point->y), 0};
30         }
31     }
32
33 }

```

Here is the call graph for this function:



4.2.3.2 get_name()

```
std::string BasePotential::get_name ( )
```

Getter for name class variable.

Returns

`std::string`

Definition at line 40 of file `base_potential.cpp`.

```

41 {
42     return this->name;
43 }

```

4.2.3.3 potential_function()

```

float BasePotential::potential_function (
    float x,
    float y )

```

Default potential function for constant zero potential.

Parameters

<i>x</i>	
<i>y</i>	

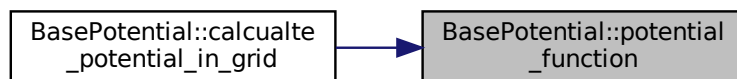
Returns

float

Definition at line 52 of file base_potential.cpp.

```
53 {  
54     return 0;  
55 }
```

Here is the caller graph for this function:



4.2.4 Member Data Documentation

4.2.4.1 name

```
std::string BasePotential::name [protected]
```

Definition at line 27 of file base_potential.h.

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

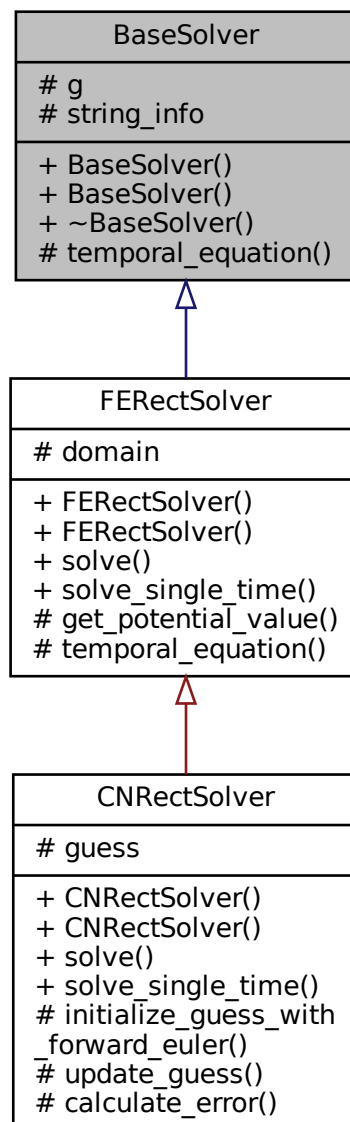
- /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/potential/base_potential.h
- /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/potential/base_potential.cpp

4.3 BaseSolver Class Reference

Base Solver for Gross Piteavskill finite difference solver.

```
#include <base_solver.h>
```

Inheritance diagram for BaseSolver:



Collaboration diagram for BaseSolver:

BaseSolver
g # string_info
+ BaseSolver() + BaseSolver() + ~BaseSolver() # temporal_equation()

Public Member Functions

- [BaseSolver](#) ()=default
- [BaseSolver](#) (float g)
Construct a new Base Solver:: Base Solver object.
- [~BaseSolver](#) ()
Destroy the Base Solver:: Base Solver object.

Protected Member Functions

- `std::complex< float > temporal_equation (int i, int j, int k)`
default temporal equation

Protected Attributes

- float [g](#)
- `std::string` [string_info](#)

4.3.1 Detailed Description

Base Solver for Gross Piteavskill finite difference solver.

Definition at line 22 of file `base_solver.h`.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 BaseSolver() [1/2]

```
BaseSolver::BaseSolver ( ) [default]
```

4.3.2.2 BaseSolver() [2/2]

```
BaseSolver::BaseSolver (
    float g_ )
```

Construct a new Base Solver:: Base Solver object.

Parameters

$g \leftrightarrow$	
$_ \leftrightarrow$	

Definition at line 18 of file base_solver.cpp.

```
19      : g(g_) {
20          this -> string_info = "Base_Solver";
21      };
```

4.3.2.3 ~BaseSolver()

```
BaseSolver::~~BaseSolver ( )
```

Destroy the Base Solver:: Base Solver object.

Definition at line 28 of file base_solver.cpp.

```
28 {};
```

4.3.3 Member Function Documentation**4.3.3.1 temporal_equation()**

```
std::complex< float > BaseSolver::temporal_equation (
    int i,
    int j,
    int k ) [protected]
```

default temporal equation

Returns

`std::complex<float> 0+0i`

Definition at line 36 of file base_solver.cpp.

```
36      {
37          return std::complex<float> {0};
38      }
```

4.3.4 Member Data Documentation

4.3.4.1 g

```
float BaseSolver::g [protected]
```

Definition at line 30 of file base_solver.h.

4.3.4.2 string_info

```
std::string BaseSolver::string_info [protected]
```

Definition at line 32 of file base_solver.h.

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

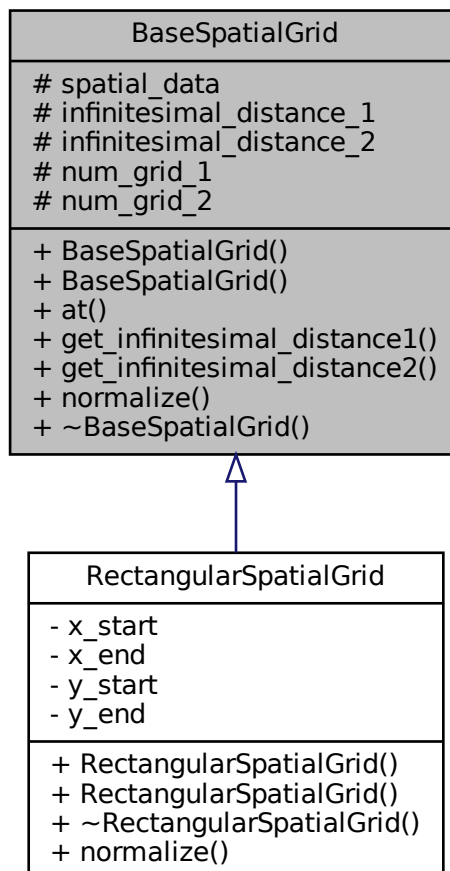
- /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/solver/[base_solver.h](#)
- /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/solver/[base_solver.cpp](#)

4.4 BaseSpatialGrid Class Reference

Base Spatial Grid class for single time step.

```
#include <base_domain.h>
```

Inheritance diagram for BaseSpatialGrid:



Collaboration diagram for BaseSpatialGrid:

BaseSpatialGrid
<pre># spatial_data # infinitesimal_distance_1 # infinitesimal_distance_2 # num_grid_1 # num_grid_2</pre>
<pre>+ BaseSpatialGrid() + BaseSpatialGrid() + at() + get_infinitesimal_distance1() + get_infinitesimal_distance2() + normalize() + ~BaseSpatialGrid()</pre>

Public Member Functions

- [BaseSpatialGrid\(\)](#)=default
- [BaseSpatialGrid](#) (int [num_grid_1](#), int [num_grid_2](#))
- [GridPoint](#) * [at](#) (int [index_1](#), int [index_2](#))
- float [get_infinitesimal_distance1](#) ()
- float [get_infinitesimal_distance2](#) ()
- void [normalize](#) ()
- [~BaseSpatialGrid](#) ()

Protected Attributes

- std::vector< std::vector< [GridPoint](#) > > [spatial_data](#)
- float [infinitesimal_distance_1](#)
- float [infinitesimal_distance_2](#)
- int [num_grid_1](#)
- int [num_grid_2](#)

4.4.1 Detailed Description

Base Spatial Grid class for single time step.

Definition at line 46 of file [base_domain.h](#).

4.4.2 Constructor & Destructor Documentation

4.4.2.1 BaseSpatialGrid() [1/2]

```
BaseSpatialGrid::BaseSpatialGrid ( ) [default]
```

4.4.2.2 BaseSpatialGrid() [2/2]

```
BaseSpatialGrid::BaseSpatialGrid (
    int num_grid_1,
    int num_grid_2 )
```

Definition at line 16 of file base_domain.cpp.

```
17 {
18     this->num_grid_1 = num_grid_1;
19     this->num_grid_2 = num_grid_2;
20     //Spatial_data: 2D matrix of grid points
21     this->spatial_data = std::vector<std::vector<GridPoint>>(num_grid_1);
22     for (auto i = 0; i < num_grid_1; ++i)
23     {
24         this->spatial_data[i] = std::vector<GridPoint>(num_grid_2);
25     }
26     this->infinitesimal_distance_1 = 1.f;
27     this->infinitesimal_distance_2 = 1.f;
28 }
```

4.4.2.3 ~BaseSpatialGrid()

```
BaseSpatialGrid::~BaseSpatialGrid ( )
```

Definition at line 51 of file base_domain.cpp.

```
52 {
53
54     for (auto i = 0; i < num_grid_1; ++i)
55     {
56         this->spatial_data[i].clear();
57         std::vector<GridPoint>().swap(this->spatial_data[i]);
58     }
59     this->spatial_data.clear();
60     std::vector<std::vector<GridPoint>>().swap(this->spatial_data);
61 }
```

4.4.3 Member Function Documentation

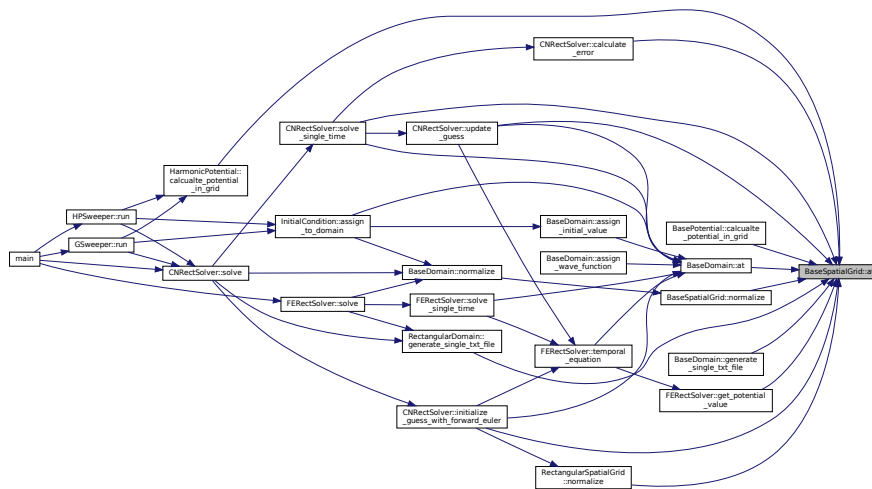
4.4.3.1 at()

```
GridPoint * BaseSpatialGrid::at (
    int index_1,
    int index_2 )
```

Definition at line 76 of file base_domain.cpp.

```
77 {
78     return &this->spatial_data[index_1 % this->num_grid_1][index_2 % this->num_grid_2];
79 }
```

Here is the caller graph for this function:



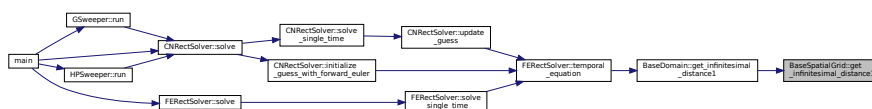
4.4.3.2 get_infinisimal_distance1()

```
float BaseSpatialGrid::get_infinisimal_distance1 ( )
```

Definition at line 64 of file base_domain.cpp.

```
65 {
66     return this->infinisimal_distance_1;
67 }
```

Here is the caller graph for this function:



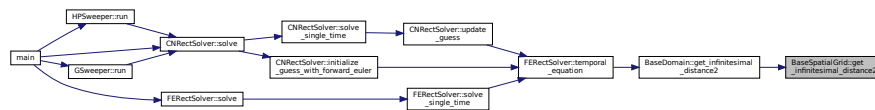
4.4.3.3 get_infinesimal_distance2()

```
float BaseSpatialGrid::get_infinesimal_distance2 ( )
```

Definition at line 70 of file base_domain.cpp.

```
71 {
72     return this->infinesimal_distance_2;
73 }
```

Here is the caller graph for this function:



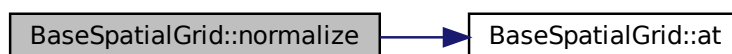
4.4.3.4 normalize()

```
void BaseSpatialGrid::normalize ( )
```

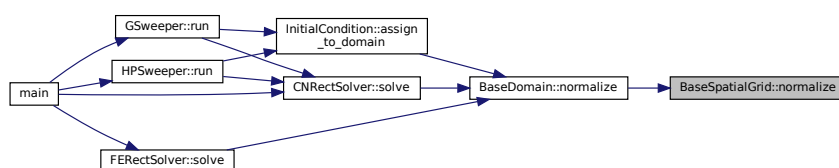
Definition at line 30 of file base_domain.cpp.

```
31 {
32     float sum = 0.;
33     for (auto i = 0; i < this->num_grid_1; ++i)
34     {
35         for (auto j = 0; j < this->num_grid_2; ++j)
36         {
37             auto wave_func = this->at(i, j)->value;
38             sum += float(std::pow(std::abs(wave_func), 2));
39         }
40     }
41     sum = std::sqrt(sum * this->infinesimal_distance_1 * this->infinesimal_distance_2);
42     for (auto i = 0; i < this->num_grid_1; ++i)
43     {
44         for (auto j = 0; j < this->num_grid_2; ++j)
45         {
46             this->at(i, j)->value /= sum;
47         }
48     }
49 }
50 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.4.4 Member Data Documentation

4.4.4.1 infinitesimal_distance_1

```
float BaseSpatialGrid::infinitesimal_distance_1 [protected]
```

Definition at line 59 of file `base_domain.h`.

4.4.4.2 infinitesimal_distance_2

```
float BaseSpatialGrid::infinitesimal_distance_2 [protected]
```

Definition at line 59 of file `base_domain.h`.

4.4.4.3 num_grid_1

```
int BaseSpatialGrid::num_grid_1 [protected]
```

Definition at line 60 of file `base_domain.h`.

4.4.4.4 num_grid_2

```
int BaseSpatialGrid::num_grid_2 [protected]
```

Definition at line 60 of file `base_domain.h`.

4.4.4.5 spatial_data

```
std::vector<std::vector<GridPoint> > BaseSpatialGrid::spatial_data [protected]
```

Definition at line 58 of file `base_domain.h`.

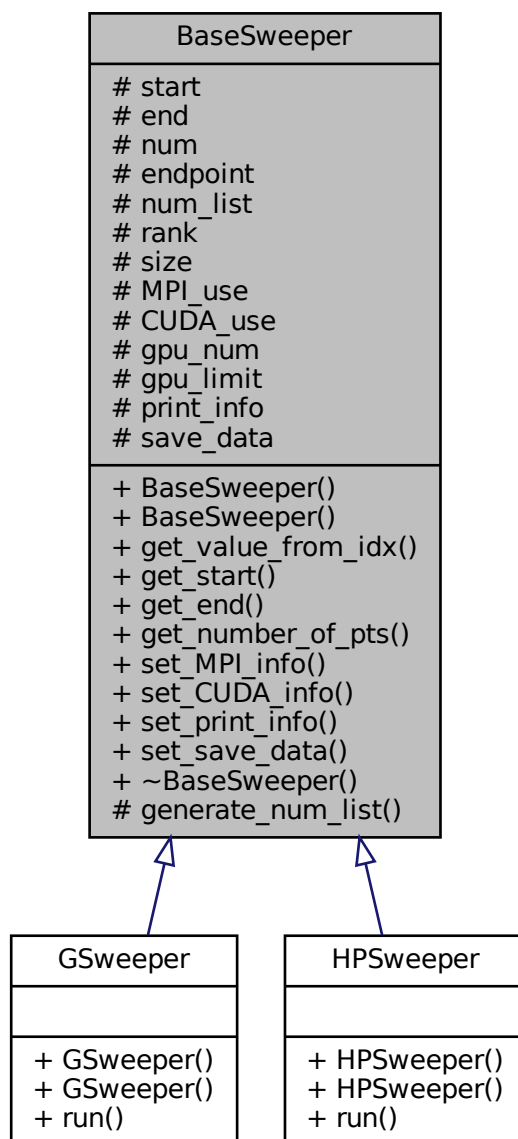
The documentation for this class was generated from the following files:

- `/home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/domain/base_domain.h`
- `/home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/domain/base_domain.cpp`

4.5 BaseSweeper Class Reference

```
#include <base_sweeper.h>
```

Inheritance diagram for BaseSweeper:



Collaboration diagram for BaseSweeper:

BaseSweeper
<pre># start # end # num # endpoint # num_list # rank # size # MPI_use # CUDA_use # gpu_num # gpu_limit # print_info # save_data</pre>
<pre>+ BaseSweeper() + BaseSweeper() + get_value_from_idx() + get_start() + get_end() + get_number_of_pts() + set_MPI_info() + set_CUDA_info() + set_print_info() + set_save_data() + ~BaseSweeper() # generate_num_list()</pre>

Public Member Functions

- [BaseSweeper](#) ()=default
- [BaseSweeper](#) (float [start](#), float [end](#), int [num](#), bool [endpoint](#)=false)
- float [get_value_from_idx](#) (int [idx](#))
- float [get_start](#) ()
- float [get_end](#) ()
- int [get_number_of_pts](#) ()
- virtual void [set_MPI_info](#) (int [rank](#), int [size](#))
- virtual void [set_CUDA_info](#) (int [gpu_num](#), int [gpu_limit](#))
- void [set_print_info](#) (bool [print_info](#))
- void [set_save_data](#) (bool [save_data](#))
- [~BaseSweeper](#) ()

Protected Member Functions

- void [generate_num_list](#) ()
generate number of points that sweep from start to end

Protected Attributes

- float `start`
- float `end`
- int `num`
- bool `endpoint`
- vector< float > `num_list`
- int `rank` =0
- int `size` =0
- bool `MPI_use`
- bool `CUDA_use`
- int `gpu_num` =1
- int `gpu_limit` =3
- bool `print_info` =true
- bool `save_data` =true

4.5.1 Detailed Description

Definition at line 7 of file `base_sweeper.h`.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 BaseSweeper() [1/2]

```
BaseSweeper::BaseSweeper ( ) [default]
```

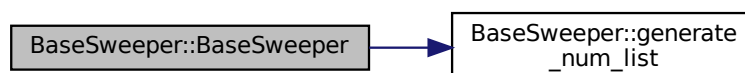
4.5.2.2 BaseSweeper() [2/2]

```
BaseSweeper::BaseSweeper (
    float start,
    float end,
    int num,
    bool endpoint = false )
```

Definition at line 5 of file `base_sweeper.cpp`.

```
5
6 start(start_), end(end_), num(num_), endpoint(endpoint_) {
7
8     this->generate_num_list();
9
10 }
```

Here is the call graph for this function:



4.5.2.3 ~BaseSweeper()

```
BaseSweeper::~~BaseSweeper ( )
```

Definition at line 63 of file base_sweeper.cpp.

```
63 {};
```

4.5.3 Member Function Documentation

4.5.3.1 generate_num_list()

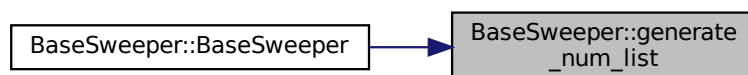
```
void BaseSweeper::generate_num_list ( ) [protected]
```

generate number of points that sweep from start to end

Definition at line 15 of file base_sweeper.cpp.

```
15 {
16     this->num_list = std::vector<float>(this->num);
17     float d = 0.;
18     if(this->endpoint){
19         d = (this -> end - this-> start ) / float(this-> num -1 );
20     }else{
21         d = (this -> end - this-> start ) / float(this-> num);
22     }
23
24     for (int i=0; i<this -> num; ++i){
25         this->num_list[i] = this ->start + d*i;
26     }
27
28 }
```

Here is the caller graph for this function:



4.5.3.2 get_end()

```
float BaseSweeper::get_end ( )
```

Definition at line 38 of file base_sweeper.cpp.

```
38 {
39     return end;
40 }
```


4.5.3.3 get_number_of_pts()

```
int BaseSweeper::get_number_of_pts ( )
```

Definition at line 41 of file base_sweeper.cpp.

```
41     {  
42         return num;  
43     }
```

4.5.3.4 get_start()

```
float BaseSweeper::get_start ( )
```

Definition at line 35 of file base_sweeper.cpp.

```
35     {  
36         return start;  
37     }
```

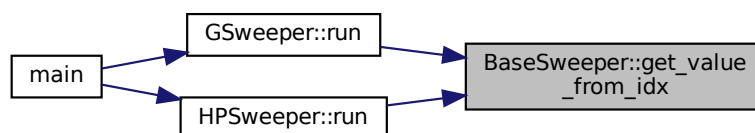
4.5.3.5 get_value_from_idx()

```
float BaseSweeper::get_value_from_idx (   
    int idx )
```

Definition at line 31 of file base_sweeper.cpp.

```
31     {  
32         return this->num_list[idx];  
33     }
```

Here is the caller graph for this function:



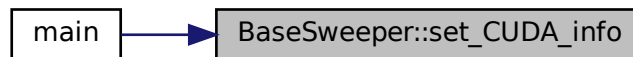
4.5.3.6 set_CUDA_info()

```
void BaseSweeper::set_CUDA_info (
    int gpu_num,
    int gpu_limit ) [virtual]
```

Definition at line 51 of file `base_sweeper.cpp`.

```
51                                     {
52     this -> CUDA_use=true;
53     this -> gpu_num = gpu_num;
54     this -> gpu_limit = gpu_limit;
55 }
```

Here is the caller graph for this function:



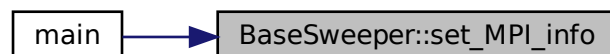
4.5.3.7 set_MPI_info()

```
void BaseSweeper::set_MPI_info (
    int rank,
    int size ) [virtual]
```

Definition at line 45 of file `base_sweeper.cpp`.

```
45                                     {
46     this -> rank = rank;
47     this -> size = size;
48     this -> MPI_use=true;
49 }
```

Here is the caller graph for this function:



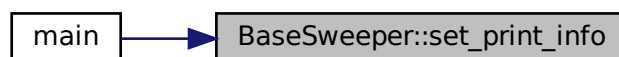
4.5.3.8 set_print_info()

```
void BaseSweeper::set_print_info (
    bool print_info )
```

Definition at line 57 of file base_sweeper.cpp.

```
57 {
58     this -> print_info = print_info;
59 }
```

Here is the caller graph for this function:



4.5.3.9 set_save_data()

```
void BaseSweeper::set_save_data (
    bool save_data )
```

Definition at line 60 of file base_sweeper.cpp.

```
60 {
61     this -> save_data = save_data;
62 }
```

Here is the caller graph for this function:



4.5.4 Member Data Documentation

4.5.4.1 CUDA_use

```
bool BaseSweeper::CUDA_use [protected]
```

Definition at line 36 of file base_sweeper.h.

4.5.4.2 end

```
float BaseSweeper::end [protected]
```

Definition at line 25 of file base_sweeper.h.

4.5.4.3 endpoint

```
bool BaseSweeper::endpoint [protected]
```

Definition at line 27 of file base_sweeper.h.

4.5.4.4 gpu_limit

```
int BaseSweeper::gpu_limit =3 [protected]
```

Definition at line 38 of file base_sweeper.h.

4.5.4.5 gpu_num

```
int BaseSweeper::gpu_num =1 [protected]
```

Definition at line 37 of file base_sweeper.h.

4.5.4.6 MPI_use

```
bool BaseSweeper::MPI_use [protected]
```

Definition at line 34 of file base_sweeper.h.

4.5.4.7 num

```
int BaseSweeper::num [protected]
```

Definition at line 26 of file base_sweeper.h.

4.5.4.8 num_list

```
vector<float> BaseSweeper::num_list [protected]
```

Definition at line 28 of file base_sweeper.h.

4.5.4.9 print_info

```
bool BaseSweeper::print_info =true [protected]
```

Definition at line 40 of file base_sweeper.h.

4.5.4.10 rank

```
int BaseSweeper::rank =0 [protected]
```

Definition at line 32 of file base_sweeper.h.

4.5.4.11 save_data

```
bool BaseSweeper::save_data =true [protected]
```

Definition at line 41 of file base_sweeper.h.

4.5.4.12 size

```
int BaseSweeper::size =0 [protected]
```

Definition at line 33 of file base_sweeper.h.

4.5.4.13 start

```
float BaseSweeper::start [protected]
```

Definition at line 24 of file base_sweeper.h.

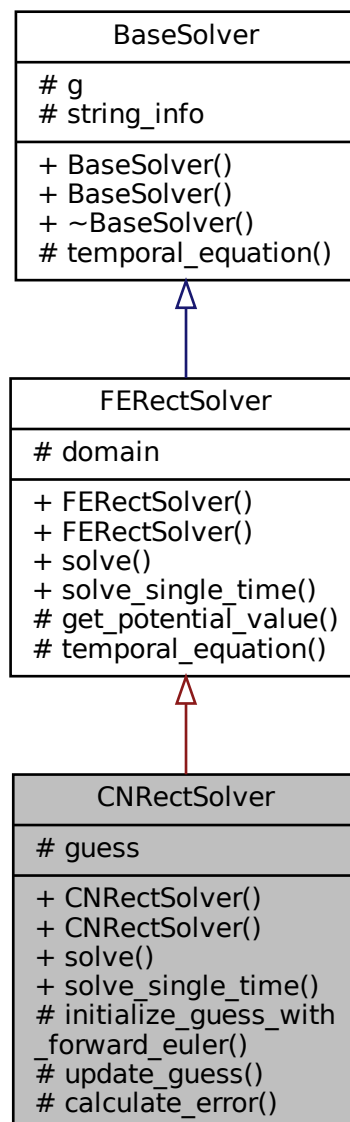
The documentation for this class was generated from the following files:

- /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/sweeper/[base_sweeper.h](#)
- /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/sweeper/[base_sweeper.cpp](#)

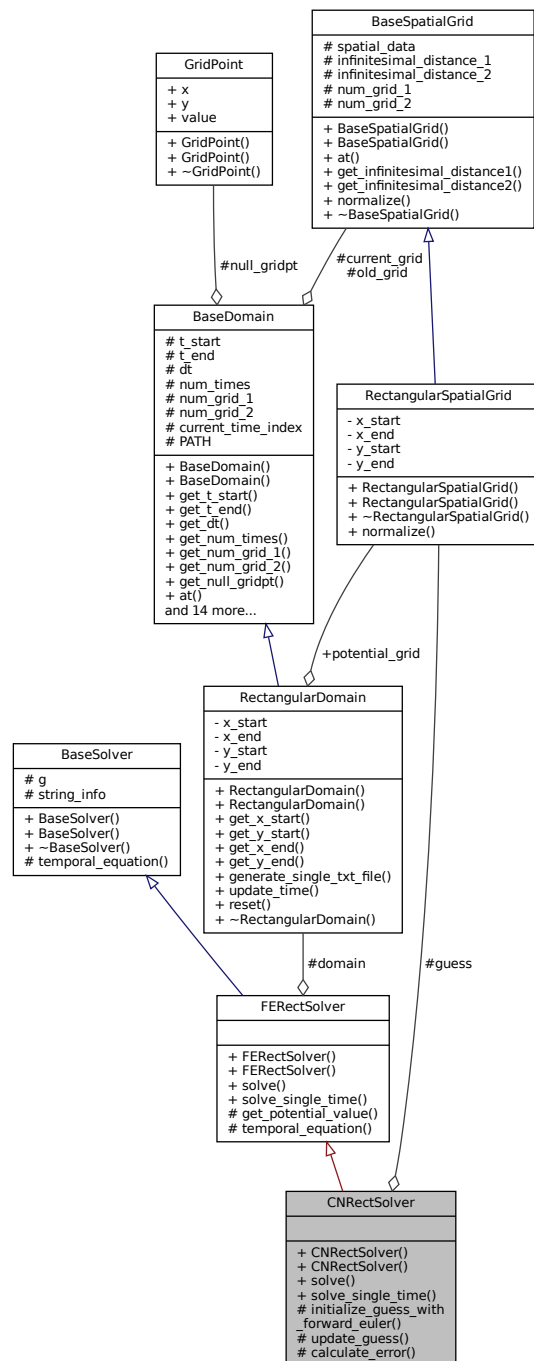
4.6 CNRectSolver Class Reference

```
#include <cn_rect_solver.h>
```

Inheritance diagram for CNRectSolver:



Collaboration diagram for CNRectSolver:



Public Member Functions

- **CNRectSolver** ()=default
- **CNRectSolver** (float `g`, **RectangularDomain** *`domain`)
Construct a new **CNRectSolver::CNRectSolver** object.
- void **solve** (float tolerance, int max_iter, std::string dir_name="", bool print_info=true, bool save_data=true)
solve equation with crank nicolson method with given max_iter and tolerance.

- void [solve_single_time](#) (int k, float tolerance, int max_iter)
Solve spatial domain for temporal index k.

Protected Member Functions

- void [initialize_guess_with_forward_euler](#) (int k)
initialize guess with forward euler method
- void [update_guess](#) (int i, int j, int k)
Update guess of the spatial(i, j), temporal k point.
- float [calculate_error](#) (int k)
Calculate L2 error between the ccurrent and previous guess.

Protected Attributes

- [RectangularSpatialGrid](#) * [guess](#)

Private Member Functions

- void [solve](#) (std::string dir_name="", bool print_info=true, bool save_data=true)
Solve Equation.
- void [solve_single_time](#) (int k)
Update k+1 th wave function value with k th values.
- float [get_potential_value](#) (int i, int j)
Get Potential value on each grid point.
- std::complex< float > [temporal_equation](#) (int i, int j, int k)
Time differential of phi.

Private Attributes

- [RectangularDomain](#) * [domain](#)
- float [g](#)
- std::string [string_info](#)

4.6.1 Detailed Description

Definition at line 24 of file `cn_rect_solver.h`.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 CNRectSolver() [1/2]

```
CNRectSolver::CNRectSolver ( ) [default]
```

4.6.2.2 CNRectSolver() [2/2]

```
CNRectSolver::CNRectSolver (
    float g,
    RectangularDomain * domain )
```

Construct a new [CNRectSolver::CNRectSolver](#) object.

Parameters

<i>g</i>	
<i>domain</i>	

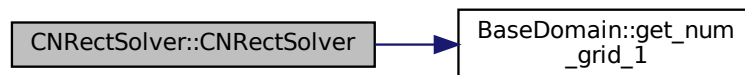
Definition at line 20 of file `cn_rect_solver.cpp`.

```

23 : FRectSolver(g, domain) //
24 {
25     // this->generate_potential_grid();
26     this->guess = new RectangularSpatialGrid(
27         this->domain->get_num_grid_1(),
28         this->domain->get_num_grid_2(),
29         this->domain->get_x_start(),
30         this->domain->get_x_end(),
31         this->domain->get_y_start(),
32         this->domain->get_y_end());
33     this->string_info = std::string{"Crank_Nicolson_serial_"};
34 };

```

Here is the call graph for this function:



4.6.3 Member Function Documentation

4.6.3.1 calculate_error()

```

float CNRectSolver::calculate_error (
    int k ) [protected]

```

Calculate L2 error between the ccurrent and previous guess.

Parameters

<i>k</i>	
----------	--

Returns

float

Definition at line 94 of file `cn_rect_solver.cpp`.

```

95 {
96     float error = 0.;
97     for (auto i = 0; i < this->domain->get_num_grid_1(); ++i)
98     {
99         for (auto j = 0; j < this->domain->get_num_grid_2(); ++j)

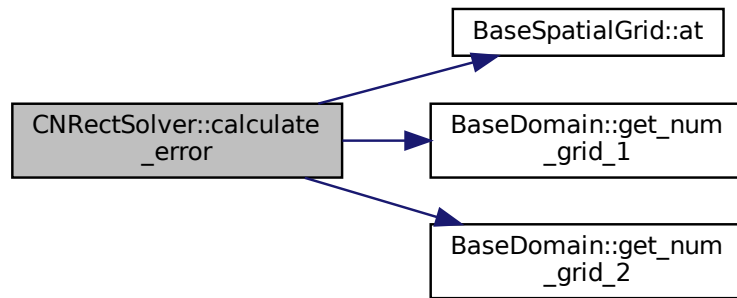
```

```

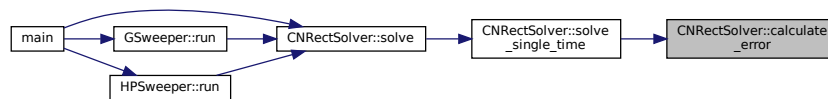
100     {
101         error += std::pow(std::abs((this->guess->at(i, j)->value - this->domain->at(i, j,
102         k)->value)), 2);
103     }
104     return error;
105 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



4.6.3.2 get_potential_value()

```

float FERectSolver::get_potential_value (
    int i,
    int j ) [protected], [inherited]

```

Get Potential value on each grid point.

Parameters

<i>i</i>	
<i>j</i>	

Returns

float

Definition at line 37 of file fe_rect_solver.cpp.

```

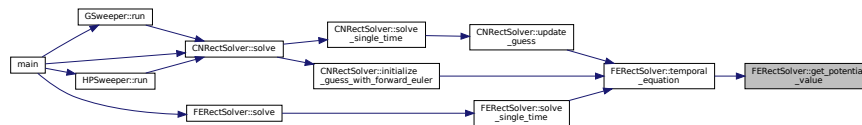
38 {
39
40     return this->domain->potential_grid->at(i, j)->value.real();
41 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



4.6.3.3 initialize_guess_with_forward_euler()

```

void CNRectSolver::initialize_guess_with_forward_euler (
    int k ) [protected]

```

initialize guess with forward euler method

Parameters

k	
-----	--

Definition at line 41 of file cn_rect_solver.cpp.

```

42 {
43     delete this->guess;
44     this->guess = new RectangularSpatialGrid(
45         this->domain->get_num_grid_1(),
46         this->domain->get_num_grid_2(),
47         this->domain->get_x_start(),
48         this->domain->get_x_end(),
49         this->domain->get_y_start(),
50         this->domain->get_y_end());
51     for (auto i = 0; i < this->domain->get_num_grid_1(); ++i)
52     {
53         for (auto j = 0; j < this->domain->get_num_grid_2(); ++j)

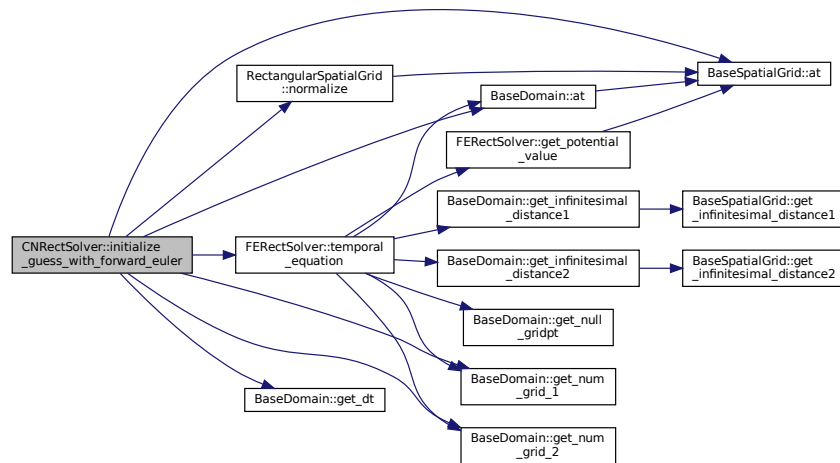
```

```

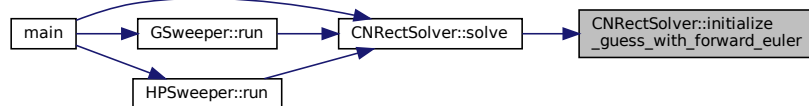
54     {
55
56         //At first, initial guess is the results of fe algorithm
57         this->domain->at(i, j, k)->value = this->domain->at(i, j, k - 1)->value +
        this->domain->get_dt() * (this->temporal_equation(i, j, k - 1));
58         this->guess->at(i, j)->value = this->domain->at(i, j, k)->value;
59     }
60 }
61 this -> guess -> normalize();
62 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



4.6.3.4 solve() [1/2]

```

void CNRectSolver::solve (
    float tolerance,
    int max_iter,
    std::string dir_name = "",
    bool print_info = true,
    bool save_data = true )

```

solve equation with crank nicolson method with given max_iter and tolerance.

Parameters

<i>tolerance</i>	
<i>max_iter</i>	
<i>dir_name</i>	
<i>print_info</i>	
<i>save_data</i>	

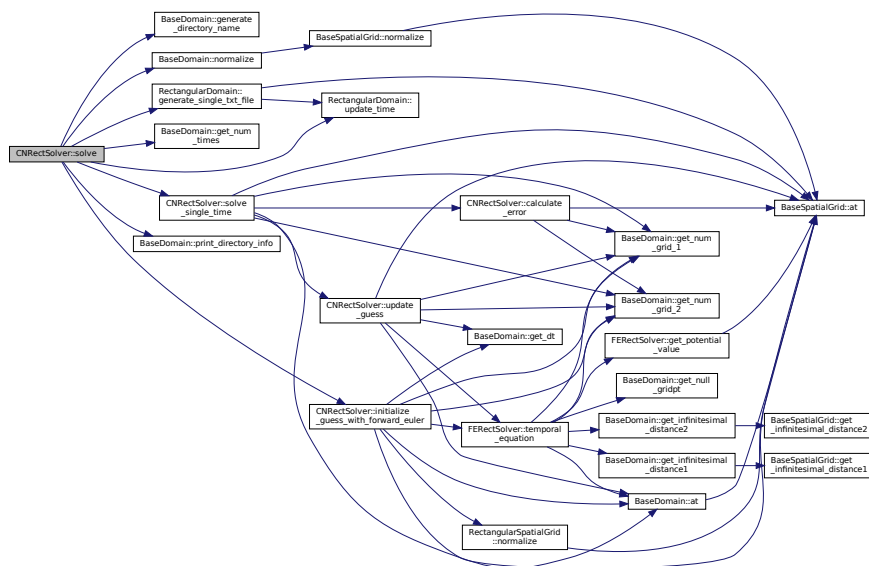
Definition at line 162 of file cn_rect_solver.cpp.

```

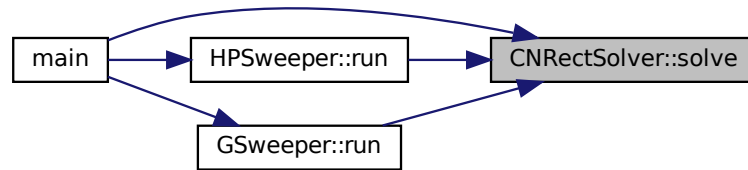
163 {
164     int time_length = this->domain->get_num_times();
165     //Save initial condition at time = start_time
166     if(save_data){
167         this->domain->generate_directory_name(this->string_info+dir_name, print_info);
168         //Save initial condition
169         this->domain->generate_single_txt_file(std::string("Solution_") + std::to_string(0));
170     }else{
171         this -> domain->update_time();
172     }
173     for (int k = 0; k < time_length-1; ++k)
174     {
175         //Update kth grid using k-1 th grid
176         // std::cout << "time step " << k << std::endl;
177
178         this->initialize_guess_with_forward_euler(k+1);
179         this->solve_single_time(k+1, tolerance, max_iter);
180         this->domain->normalize(k+1);
181         if(save_data){
182             this->domain->generate_single_txt_file(std::string("Solution_") + std::to_string(k+1));
183         }
184         else{
185             this->domain->update_time();
186         }
187     }
188     if(print_info){
189         this->domain->print_directory_info();
190     }
191 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



4.6.3.5 solve() [2/2]

```

void FERectSolver::solve (
    std::string dir_name = "",
    bool print_info = true,
    bool save_data = true ) [inherited]

```

Solve Equation.

Parameters

<i>dir_name</i>	
<i>print_info</i>	
<i>save_data</i>	

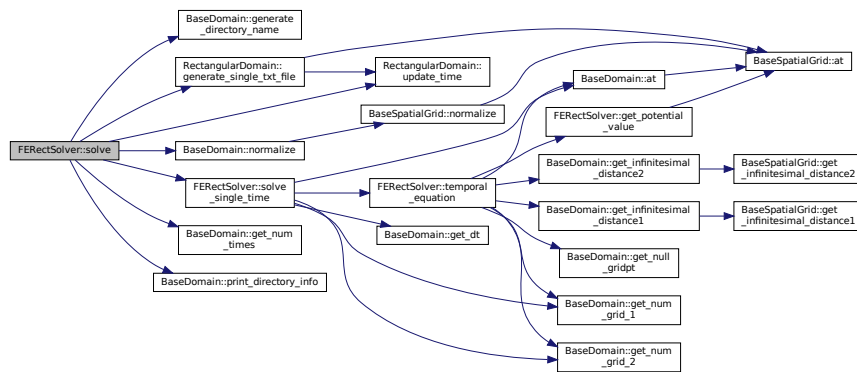
Definition at line 116 of file `fe_rect_solver.cpp`.

```

117 {
118     int time_length = this->domain->get_num_times();
119     if(save_data){
120         this->domain->generate_directory_name(this->string_info+dir_name, print_info);
121         //Save initial condition
122         this->domain->generate_single_txt_file(std::string("Solution_") + std::to_string(0));
123     }else{
124         this -> domain->update_time();
125     }
126
127     for (int k = 0; k < time_length - 1; ++k)
128     {
129         this->solve_single_time(k);
130         this->domain->normalize(k + 1);
131         if(save_data){
132             this->domain->generate_single_txt_file(std::string("Solution_") + std::to_string(k + 1));
133         }
134         else{
135             this->domain->update_time();
136         }
137     }
138     if(print_info){
139         this->domain->print_directory_info();
140     }
141 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



4.6.3.6 solve_single_time() [1/2]

```
void FERectSolver::solve_single_time (
    int k ) [inherited]
```

Update k+1 th wave function value with k th values.

Parameters

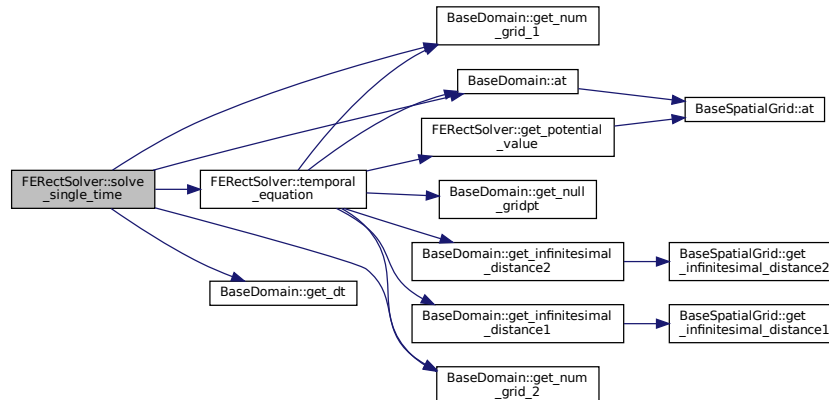
<i>k</i>	
----------	--

Definition at line 95 of file `fe_rect_solver.cpp`.

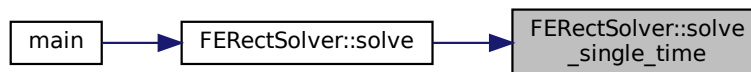
```

96 {
97     int Nx = this->domain->get_num_grid_1();
98     int Ny = this->domain->get_num_grid_2();
99     float dt = this->domain->get_dt();
100     for (int i = 0; i < Nx; ++i)
101     {
102         for (int j = 0; j < Ny; ++j)
103         {
104             (this->domain->at(i, j, k + 1)->value) = this->domain->at(i, j, k)->value + dt *
105             this->temporal_equation(i, j, k);
106         }
107     }
108 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.6.3.7 solve_single_time() [2/2]

```

void CNRectSolver::solve_single_time (
    int k,
    float tolerance,
    int max_iter )
  
```

Solve spatial domain for temporal index k.

Parameters

<i>k</i>	
<i>tolerance</i>	
<i>max_iter</i>	

Definition at line 114 of file cn_rect_solver.cpp.

```

115 {
116     float error = 1.;
117     bool converged = false;
118     int converged_step = 0;
119     for (auto iter = 0; iter < max_iter; ++iter)
  
```

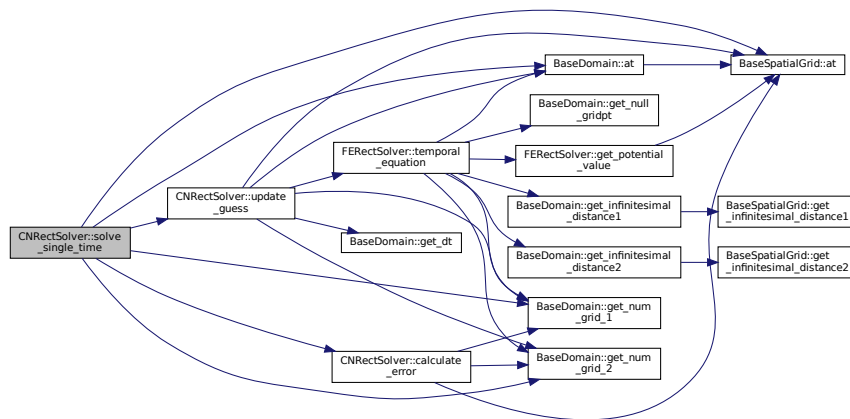


```

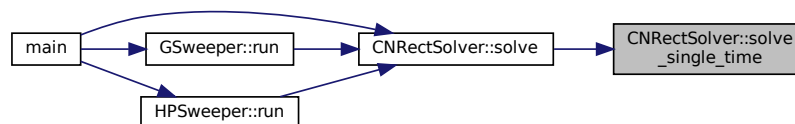
120     {
121         if (error < tolerance)
122         {
123             converged = true;
124             converged_step = iter - 1;
125             break;
126         }
127         //For each point, update wave function
128         for (auto i = 0; i < this->domain->get_num_grid_1(); ++i)
129         {
130             for (auto j = 0; j < this->domain->get_num_grid_2(); ++j)
131             {
132                 this->domain->at(i, j, k)->value = this->guess->at(i, j)->value;
133             }
134         }
135
136         //for each point, update predicted value
137         for (auto i = 0; i < this->domain->get_num_grid_1(); ++i)
138         {
139             for (auto j = 0; j < this->domain->get_num_grid_2(); ++j)
140             {
141                 update_guess(i, j, k);
142             }
143         }
144         error = this->calculate_error(k);
145     }
146 }
147 if (!converged)
148 {
149     std::cout << "At time " << k << " Converged failed with error = " << error << std::endl;
150 }
151 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



4.6.3.8 temporal_equation()

```
std::complex< float > FERectSolver::temporal_equation (
    int i,
    int j,
    int k ) [protected], [inherited]
```

Time differential of phi.

Parameters

<i>i</i>	index for x
<i>j</i>	index for y
<i>k</i>	index for time(t)

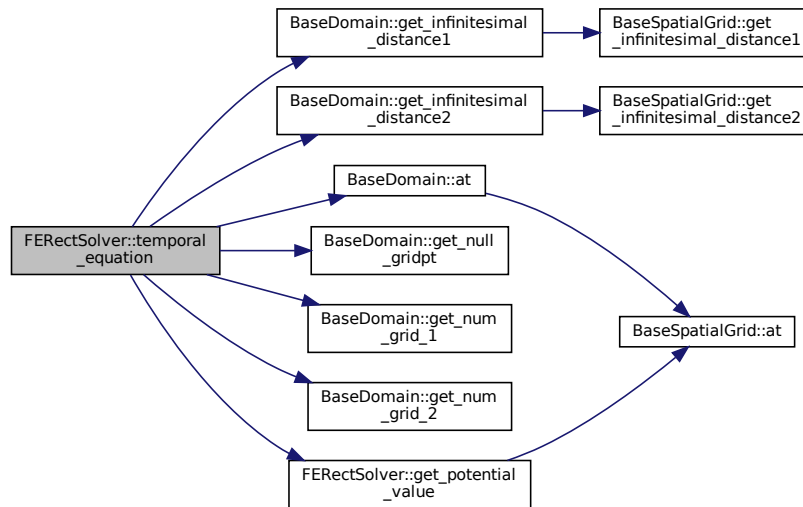
Returns

std::complex<float> time differential at x, y, t

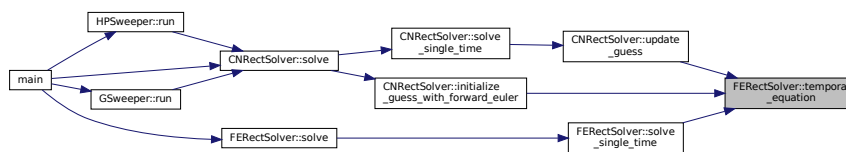
Definition at line 51 of file fe_rect_solver.cpp.

```
52 {
53     //Use five stencil method
54     auto point_data = this->domain->at(i, j, k);
55
56     //l,r,d,u denotes left, right, down, up value
57     //Check boundary
58     auto point_data_l = this->domain->at(i - 1, j, k);
59     if (i <= 0)
60         point_data_l = this->domain->get_null_gridpt();
61     auto point_data_d = this->domain->at(i, j - 1, k);
62     if (j <= 0)
63         point_data_d = this->domain->get_null_gridpt();
64     auto point_data_r = this->domain->at(i + 1, j, k);
65     if (i >= (this->domain->get_num_grid_1() - 1))
66         point_data_r = this->domain->get_null_gridpt();
67     auto point_data_u = this->domain->at(i, j + 1, k);
68     if (j >= (this->domain->get_num_grid_2() - 1))
69         point_data_u = this->domain->get_null_gridpt();
70
71     //potential at x, y
72     float V_ij = this->get_potential_value(i, j);
73     //this->potential_func(point_data->x, point_data->y);
74
75     //g * |psi(x,y)|^2
76     float additional_term = (this->g) * (std::abs(point_data->value)) * (std::abs(point_data->value));
77
78     //Set infinitesimal value
79     float dx = this->domain->get_infinitesimal_distance1();
80     float dy = this->domain->get_infinitesimal_distance2();
81     //df denote time differential of dt (d(psi)/dt)
82     // = (laplace - V-g|psi|^2) psi
83     std::complex<float> df =
84         ((point_data_r->value) + (point_data_l->value) - (point_data->value) * std::complex<float>{2}) /
85         (std::complex<float>{dx * dx}) + ((point_data_u->value) + (point_data_d->value) - (point_data->value)
86         * std::complex<float>{2}) / (std::complex<float>{dy * dy}) - (V_ij + additional_term) *
87         (point_data->value);
88     df *= std::complex<float>{0, 1};
89
90     return df;
91 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.6.3.9 update_guess()

```

void CNRectSolver::update_guess (
    int i,
    int j,
    int k ) [protected]
  
```

Update guess of the spatial(i, j), temporal k point.

Parameters

<i>i</i>	
<i>j</i>	
<i>k</i>	

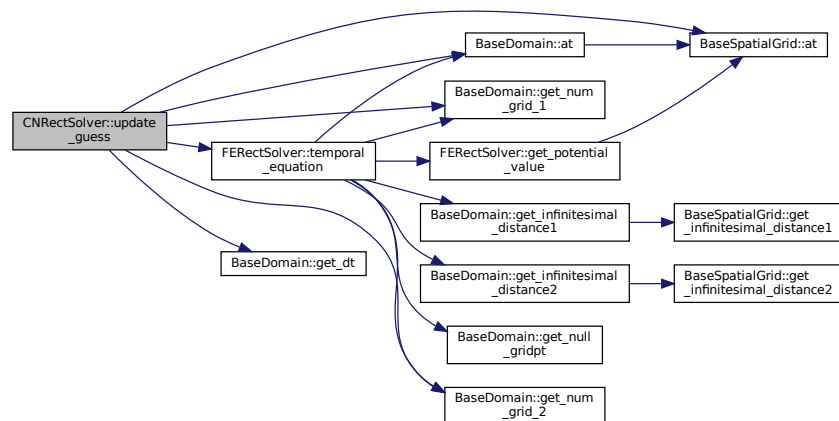
Definition at line 71 of file `cn_rect_solver.cpp`.

```

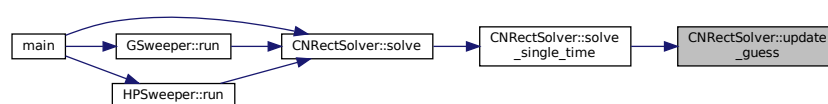
72 {
73     for (auto i = 0; i < this->domain->get_num_grid_1(); ++i)
74     {
75         for (auto j = 0; j < this->domain->get_num_grid_2(); ++j)
76         {
77             this->guess->at(i, j)->value = (
78                 0.99f * this->guess->at(i, j)->value +
79                 0.01f * (
80                     this->domain->at(i, j, k - 1)->value +
81                     std::complex<float>{0.5, 0.} * this->domain->get_dt() * (
82                         this->temporal_equation(i, j, k - 1) +
83                         this->temporal_equation(i, j, k))); // this->temporal_equation_from_guess(i,
84             j));
85         }
86     }

```

Here is the call graph for this function:



Here is the caller graph for this function:



4.6.4 Member Data Documentation

4.6.4.1 domain

`RectangularDomain*` `FERectSolver::domain` [protected], [inherited]

Definition at line 39 of file `fe_rect_solver.h`.

4.6.4.2 g

```
float BaseSolver::g [protected], [inherited]
```

Definition at line 30 of file base_solver.h.

4.6.4.3 guess

```
RectangularSpatialGrid* CNRectSolver::guess [protected]
```

Definition at line 35 of file cn_rect_solver.h.

4.6.4.4 string_info

```
std::string BaseSolver::string_info [protected], [inherited]
```

Definition at line 32 of file base_solver.h.

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

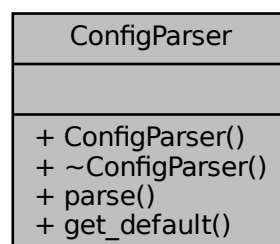
- [/home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/solver/serial_solver/crank_nicolson/cn_rect_solver.h](#)
- [/home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/solver/serial_solver/crank_nicolson/cn_rect_solver.cpp](#)

4.7 ConfigParser Class Reference

Configuration parser.

```
#include <config_parser.h>
```

Collaboration diagram for ConfigParser:



Public Member Functions

- [ConfigParser](#) ()=default
- virtual [~ConfigParser](#) ()=default

Static Public Member Functions

- static [Parameters](#) [parse](#) (std::string config_name, std::string filename)
Parse configuration for the calculation.
- static [Parameters](#) [get_default](#) ()
Setup parameters with default values.

4.7.1 Detailed Description

Configuration parser.

Definition at line 27 of file config_parser.h.

4.7.2 Constructor & Destructor Documentation

4.7.2.1 ConfigParser()

```
ConfigParser::ConfigParser ( ) [default]
```

4.7.2.2 ~ConfigParser()

```
virtual ConfigParser::~~ConfigParser ( ) [virtual], [default]
```

4.7.3 Member Function Documentation

4.7.3.1 get_default()

`Parameters` `ConfigParser::get_default () [static]`

Setup parameters with default values.

Returns

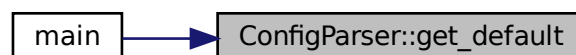
`Parameters`

Definition at line 494 of file `config_parser.cpp`.

```

495 {
496     auto parameters = Parameters();
497     parameters.config_name = "default";
498     MainParameters main_parameters;
499     DomainParameters domain_parameters;
500     InitialConditionParameters init_parameters;
501     EquationParameters equation_parameters;
502     SolverParameters solver_parameters;
503
504     main_parameters.calculation_type = "single";
505     domain_parameters.domain_type = "rectangular";
506     domain_parameters.n_x = 256;
507     domain_parameters.n_y = 256;
508     domain_parameters.n_time = 5;
509     domain_parameters.time_start = 0.;
510     domain_parameters.time_end = 0.01;
511     domain_parameters.spatial_parameters["x_start"] = -10.;
512     domain_parameters.spatial_parameters["x_end"] = 10.;
513     domain_parameters.spatial_parameters["y_start"] = -10.;
514     domain_parameters.spatial_parameters["y_end"] = 10.;
515
516     init_parameters.init_cond_type = "singlegaussian";
517     init_parameters.init_cond_parameters["sigma_x"] = 1.;
518     init_parameters.init_cond_parameters["sigma_y"] = 1.;
519     init_parameters.init_cond_parameters["center_x"] = 0.;
520     init_parameters.init_cond_parameters["center_y"] = 0.;
521
522     equation_parameters.potential_type = "harmonic";
523     equation_parameters.g = -1;
524     equation_parameters.potential_parameters["omega_x"] = 3.;
525     equation_parameters.potential_parameters["omega_y"] = 5.;
526
527     solver_parameters.method = "cranknicolson";
528     solver_parameters.run_parallel = true;
529     solver_parameters.print_info = false;
530     solver_parameters.save_data = false;
531     solver_parameters.solver_parameters["converge_crit"] = 1e-11;
532     solver_parameters.int_parameters["max_iter"] = 1001;
533     solver_parameters.int_parameters["cuda_device"] = 0;
534
535     parameters.main_parameters = main_parameters;
536     parameters.domain_parameters = domain_parameters;
537     parameters.init_cond_parameters = init_parameters;
538     parameters.equation_parameters = equation_parameters;
539     parameters.solver_parameters = solver_parameters;
540     return parameters;
541 }
```

Here is the caller graph for this function:



4.7.3.2 parse()

```
Parameters ConfigParser::parse (
    std::string config_name,
    std::string filename ) [static]
```

Parse configuration for the calculation.

Parameters

<i>config_name</i>	configuration name
<i>filename</i>	configuration file name

Returns

Parsed parameters

Definition at line 21 of file config_parser.cpp.

```
21
22     auto parameters = Parameters();
23     parameters.config_name = config_name;
24     // -1 for outside, 0 for main, 1 for domain, 2 for initial condition, 3 for equation, 4 for solver
25     int mode = -1;
26     MainParameters main_parameters;
27     DomainParameters domain_parameters;
28     InitialConditionParameters init_parameters;
29     EquationParameters equation_parameters;
30     SolverParameters solver_parameters;
31     bool filled_parallel = false;
32
33     std::ifstream file(filename.c_str());
34     if (file.is_open()) // If file is opened
35     {
36         std::string line;
37         while (std::getline(file, line)) // while not EOF
38         {
39             // remove all blanks in the line
40             line.erase(std::remove(line.begin(), line.end(), ' '), line.end());
41             // replace all lines with lower case
42             std::transform(line.begin(), line.end(), line.begin(),
43                 [](unsigned char c)
44                 { return std::tolower(c); });
45
46             // skip the line when whole line is blank
47             if (line == ""){
48                 continue;
49             }
50             // skip the line for the comment
51             else if (line.rfind("#", 0) != std::string::npos)
52             {
53                 continue;
54             }
55             // Divide the configuration branches (0 for main, 1 for domain, 2 for initial condition, 3
56             for equation, 4 for solver)
57             else if (line.rfind("[mainconfiguration]", 0) != std::string::npos)
58             {
59                 mode = 0;
60             }
61             else if (line.rfind("[domainconfiguration]", 0) != std::string::npos)
62             {
63                 mode = 1;
64             }
65             else if (line.rfind("[initialconditionconfiguration]", 0) != std::string::npos)
66             {
67                 mode = 2;
68             }
69             else if (line.rfind("[equationconfiguration]", 0) != std::string::npos)
70             {
71                 mode = 3;
72             }
73             else if (line.rfind("[solverconfiguration]", 0) != std::string::npos)
74             {
75                 mode = 4;
76             }
77         }
78     }
```



```

76         // Do action for each branches
77     else{
78         // Handle exception
79         if (mode == -1){
80             std::cerr << "Unexpected Configuration File" << std::endl;
81         }
82         // Main branch
83     else if (mode == 0){
84         std::vector<std::string> dump;
85         std::string tmp;
86         std::stringstream string_stream(line);
87         if (line.rfind("type", 0) != std::string::npos)
88         {
89             while (std::getline(string_stream, tmp, '='))
90             {
91                 dump.push_back(tmp);
92             }
93             if (dump.size() != 2)
94             {
95                 std::cerr << "Unexpected Configuration File" << std::endl;
96             }
97             main_parameters.calculation_type = dump[1];
98         }
99     else{
100         while (std::getline(string_stream, tmp, '='))
101         {
102             dump.push_back(tmp);
103         }
104         if (dump.size() != 2)
105         {
106             std::cerr << "Unexpected Configuration File" << std::endl;
107         }
108         if (dump[0] == "sweep_start")
109         {
110             main_parameters.float_parameters[dump[0]] =
111             (float)std::atof(dump[1].c_str());
112         }
113         else if (dump[0] == "sweep_end")
114         {
115             main_parameters.float_parameters[dump[0]] =
116             (float)std::atof(dump[1].c_str());
117         }
118         else if (dump[0] == "sweep_count")
119         {
120             main_parameters.int_parameters[dump[0]] = std::atoi(dump[1].c_str());
121         }
122         else if (dump[0] == "end_point")
123         {
124             if (dump[1] == "true")
125             {
126                 main_parameters.int_parameters[dump[0]] = (int)true;
127             }
128             else if (dump[1] == "false")
129             {
130                 main_parameters.int_parameters[dump[0]] = (int>false;
131             }
132             else
133             {
134                 std::cerr << "Unexpected Configuration File" << std::endl;
135             }
136         }
137         else if (dump[0] == "mpi_use")
138         {
139             if (dump[1] == "true")
140             {
141                 main_parameters.int_parameters[dump[0]] = (int)true;
142             }
143             else if (dump[1] == "false")
144             {
145                 main_parameters.int_parameters[dump[0]] = (int>false;
146             }
147             else
148             {
149                 std::cerr << "Unexpected Configuration File" << std::endl;
150             }
151         }
152         else if (dump[0] == "cuda_use")
153         {
154             if (dump[1] == "true")
155             {
156                 main_parameters.int_parameters[dump[0]] = (int)true;
157             }
158             else if (dump[1] == "false")
159             {
160                 main_parameters.int_parameters[dump[0]] = (int>false;
161             }
162             else

```

```

161         {
162             std::cerr << "Unexpected Configuration File" << std::endl;
163         }
164     }
165     else if (dump[0] == "gpu_count")
166     {
167         main_parameters.int_parameters[dump[0]] = std::atoi(dump[1].c_str());
168     }
169     else if (dump[0] == "calculation_per_gpu")
170     {
171         main_parameters.int_parameters[dump[0]] = std::atoi(dump[1].c_str());
172     }
173     else
174     {
175         std::cerr << "Unexpected Configuration File" << std::endl;
176     }
177 }
178
179 // domain branch
180 else if(mode == 1){
181     std::vector<std::string> dump;
182     std::string tmp;
183     std::stringstream string_stream(line);
184     if (line.rfind("type", 0) != std::string::npos)
185     {
186         while (std::getline(string_stream, tmp, '='))
187         {
188             dump.push_back(tmp);
189         }
190         if (dump.size() != 2){
191             std::cerr << "Unexpected Configuration File" << std::endl;
192         }
193         domain_parameters.domain_type = dump[1];
194     }
195     else{
196         while (std::getline(string_stream, tmp, '='))
197         {
198             dump.push_back(tmp);
199         }
200         if (dump.size() != 2)
201         {
202             std::cerr << "Unexpected Configuration File" << std::endl;
203         }
204
205         if (domain_parameters.domain_type != ""){
206             while (std::getline(string_stream, tmp, '='))
207             {
208                 dump.push_back(tmp);
209             }
210             if (dump.size() != 2)
211             {
212                 std::cerr << "Unexpected Configuration File" << std::endl;
213             }
214
215             if (dump[0] == "n_x"){
216                 domain_parameters.n_x = std::atoi(dump[1].c_str());
217             }
218             else if (dump[0] == "n_y"){
219                 domain_parameters.n_y = std::atoi(dump[1].c_str());
220             }
221             else if (dump[0] == "n_time"){
222                 domain_parameters.n_time = std::atoi(dump[1].c_str());
223             }
224             else if (dump[0] == "time_start"){
225                 domain_parameters.time_start = (float)std::atof(dump[1].c_str());
226             }
227             else if (dump[0] == "time_end"){
228                 domain_parameters.time_end = (float)std::atof(dump[1].c_str());
229             }
230             else{
231                 domain_parameters.spatial_parameters[dump[0]] =
232                 (float)std::atof(dump[1].c_str());
233             }
234             else{
235                 std::cerr << "Unexpected Configuration File" << std::endl;
236             }
237         }
238     }
239     // initial condition branches
240     else if(mode == 2){
241         std::vector<std::string> dump;
242         std::string tmp;
243         std::stringstream string_stream(line);
244         if (line.rfind("type", 0) != std::string::npos)
245         {
246             while (std::getline(string_stream, tmp, '='))

```

```

247         {
248             dump.push_back(tmp);
249         }
250         if (dump.size() != 2)
251         {
252             std::cerr << "Unexpected Configuration File" << std::endl;
253         }
254         init_parameters.init_cond_type = dump[1];
255     }
256     else
257     {
258         while (std::getline(string_stream, tmp, '='))
259         {
260             dump.push_back(tmp);
261         }
262         if (dump.size() != 2)
263         {
264             std::cerr << "Unexpected Configuration File" << std::endl;
265         }
266         if (init_parameters.init_cond_type != "")
267         {
268             init_parameters.init_cond_parameters[dump[0]] =
269             (float)std::atof(dump[1].c_str());
270         }
271         else
272         {
273             std::cerr << "Unexpected Configuration File" << std::endl;
274         }
275     }
276 }
277 // equation branch
278 else if(mode == 3){
279     std::vector<std::string> dump;
280     std::string tmp;
281     std::stringstream string_stream(line);
282     if (line.rfind("type", 0) != std::string::npos)
283     {
284         while (std::getline(string_stream, tmp, '='))
285         {
286             dump.push_back(tmp);
287         }
288         if (dump.size() != 2)
289         {
290             std::cerr << "Unexpected Configuration File" << std::endl;
291         }
292         equation_parameters.potential_type = dump[1];
293     }
294     else if (line.rfind("g", 0) != std::string::npos)
295     {
296         while (std::getline(string_stream, tmp, '='))
297         {
298             dump.push_back(tmp);
299         }
300         if (dump.size() != 2)
301         {
302             std::cerr << "Unexpected Configuration File" << std::endl;
303         }
304         equation_parameters.g = std::atof(dump[1].c_str());
305     }
306     else
307     {
308         while (std::getline(string_stream, tmp, '='))
309         {
310             dump.push_back(tmp);
311         }
312         if (dump.size() != 2)
313         {
314             std::cerr << "Unexpected Configuration File" << std::endl;
315         }
316         if (equation_parameters.potential_type != "")
317         {
318             equation_parameters.potential_parameters[dump[0]] =
319             std::atof(dump[1].c_str());
320         }
321         else
322         {
323             std::cerr << "Unexpected Configuration File" << std::endl;
324         }
325     }
326 }
327 // solver branch
328 else if (mode == 4){
329     std::vector<std::string> dump;
330     std::string tmp;
331     std::stringstream string_stream(line);

```

```

332
333 // Get Method
334 if (line.rfind("method", 0) != std::string::npos)
335 {
336     while (std::getline(string_stream, tmp, '='))
337     {
338         dump.push_back(tmp);
339     }
340     if (dump.size() != 2)
341     {
342         std::cerr << "Unexpected Configuration File" << std::endl;
343     }
344     solver_parameters.method = dump[1];
345 }
346
347 // Get Parallel
348 else if (line.rfind("parallel", 0) != std::string::npos)
349 {
350     while (std::getline(string_stream, tmp, '='))
351     {
352         dump.push_back(tmp);
353     }
354     if (dump.size() != 2)
355     {
356         std::cerr << "Unexpected Configuration File" << std::endl;
357     }
358
359     if (dump[1].rfind("true", 0) != std::string::npos)
360     {
361         solver_parameters.run_parallel = true;
362         filled_parallel = true;
363     }
364     else if (dump[1].rfind("false", 0) != std::string::npos)
365     {
366         solver_parameters.run_parallel = false;
367         filled_parallel = true;
368     }
369     else{
370         std::cerr << "Unexpected Configuration File" << std::endl;
371     }
372 }
373
374 // Get Print Info
375 else if (line.rfind("print_info", 0) != std::string::npos)
376 {
377     while (std::getline(string_stream, tmp, '='))
378     {
379         dump.push_back(tmp);
380     }
381     if (dump.size() != 2)
382     {
383         std::cerr << "Unexpected Configuration File" << std::endl;
384     }
385     if (dump[1].rfind("true", 0) != std::string::npos)
386     {
387         solver_parameters.print_info = true;
388     }
389     else if (dump[1].rfind("false", 0) != std::string::npos)
390     {
391         solver_parameters.print_info = false;
392     }
393     else
394     {
395         std::cerr << "Unexpected Configuration File" << std::endl;
396     }
397 }
398
399 // Get Save Data
400 else if (line.rfind("save_data", 0) != std::string::npos)
401 {
402     while (std::getline(string_stream, tmp, '='))
403     {
404         dump.push_back(tmp);
405     }
406     if (dump.size() != 2)
407     {
408         std::cerr << "Unexpected Configuration File" << std::endl;
409     }
410     if (dump[1].rfind("true", 0) != std::string::npos)
411     {
412         solver_parameters.save_data = true;
413     }
414     else if (dump[1].rfind("false", 0) != std::string::npos)
415     {
416         solver_parameters.save_data = false;
417     }
418     else

```

```

419         {
420             std::cerr << "Unexpected Configuration File" << std::endl;
421         }
422     }
423
424     // Get Cuda Device
425     else if (line.rfind("cuda_device", 0) != std::string::npos)
426     {
427         while (std::getline(string_stream, tmp, '='))
428         {
429             dump.push_back(tmp);
430         }
431         if (dump.size() != 2)
432         {
433             std::cerr << "Unexpected Configuration File" << std::endl;
434         }
435         solver_parameters.int_parameters[dump[0]] = std::atoi(dump[1].c_str());
436     }
437
438     // Get max iter
439     else if (line.rfind("max_iter", 0) != std::string::npos)
440     {
441         while (std::getline(string_stream, tmp, '='))
442         {
443             dump.push_back(tmp);
444         }
445         if (dump.size() != 2)
446         {
447             std::cerr << "Unexpected Configuration File" << std::endl;
448         }
449         solver_parameters.int_parameters[dump[0]] = std::atoi(dump[1].c_str());
450     }
451
452     else
453     {
454         while (std::getline(string_stream, tmp, '='))
455         {
456             dump.push_back(tmp);
457         }
458         if (dump.size() != 2)
459         {
460             std::cerr << "Unexpected Configuration File" << std::endl;
461         }
462         if ( (solver_parameters.method != "") && filled_parallel)
463         {
464             solver_parameters.solver_parameters[dump[0]] = std::atof(dump[1].c_str());
465         }
466         else
467         {
468             std::cerr << "Unexpected Configuration File" << std::endl;
469         }
470     }
471
472     }
473     else{
474         std::cerr << "Unexpected Configuration File" << std::endl;
475     }
476 }
477 }
478 }
479 // setup parameters
480 parameters.main_parameters = main_parameters;
481 parameters.domain_parameters = domain_parameters;
482 parameters.init_cond_parameters = init_parameters;
483 parameters.equation_parameters = equation_parameters;
484 parameters.solver_parameters = solver_parameters;
485 return parameters;
486 }

```

Here is the caller graph for this function:



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

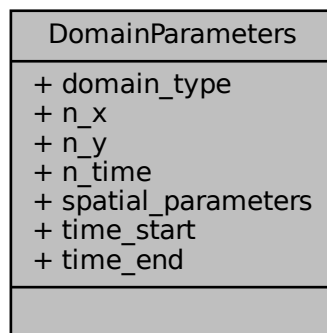
- [/home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/configuration/config_parser.h](#)
- [/home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/configuration/config_parser.cpp](#)

4.8 DomainParameters Class Reference

Domain branch parameters containing information about domain.

```
#include <parameters.h>
```

Collaboration diagram for DomainParameters:



Public Attributes

- `std::string domain_type = std::string("")`
- `int n_x`
- `int n_y`
- `int n_time`
- `std::map< std::string, float > spatial_parameters`
- `float time_start`
- `float time_end`

4.8.1 Detailed Description

Domain branch parameters containing information about domain.

Definition at line 33 of file `parameters.h`.

4.8.2 Member Data Documentation

4.8.2.1 domain_type

```
std::string DomainParameters::domain_type = std::string("")
```

Definition at line 36 of file parameters.h.

4.8.2.2 n_time

```
int DomainParameters::n_time
```

Definition at line 37 of file parameters.h.

4.8.2.3 n_x

```
int DomainParameters::n_x
```

Definition at line 37 of file parameters.h.

4.8.2.4 n_y

```
int DomainParameters::n_y
```

Definition at line 37 of file parameters.h.

4.8.2.5 spatial_parameters

```
std::map<std::string, float> DomainParameters::spatial_parameters
```

Definition at line 38 of file parameters.h.

4.8.2.6 time_end

```
float DomainParameters::time_end
```

Definition at line 39 of file parameters.h.

4.8.2.7 time_start

```
float DomainParameters::time_start
```

Definition at line 39 of file parameters.h.

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

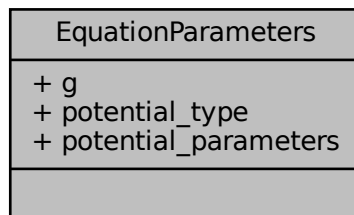
- /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/configuration/[parameters.h](#)

4.9 EquationParameters Class Reference

Equation branch parameters containing information about equation.

```
#include <parameters.h>
```

Collaboration diagram for EquationParameters:



Public Attributes

- float [g](#)
- std::string [potential_type](#) = std::string("")
- std::map< std::string, float > [potential_parameters](#)

4.9.1 Detailed Description

Equation branch parameters containing information about equation.

Definition at line 57 of file parameters.h.

4.9.2 Member Data Documentation

4.9.2.1 g

```
float EquationParameters::g
```

Definition at line 60 of file parameters.h.

4.9.2.2 potential_parameters

```
std::map<std::string, float> EquationParameters::potential_parameters
```

Definition at line 62 of file parameters.h.

4.9.2.3 potential_type

```
std::string EquationParameters::potential_type = std::string("")
```

Definition at line 61 of file parameters.h.

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

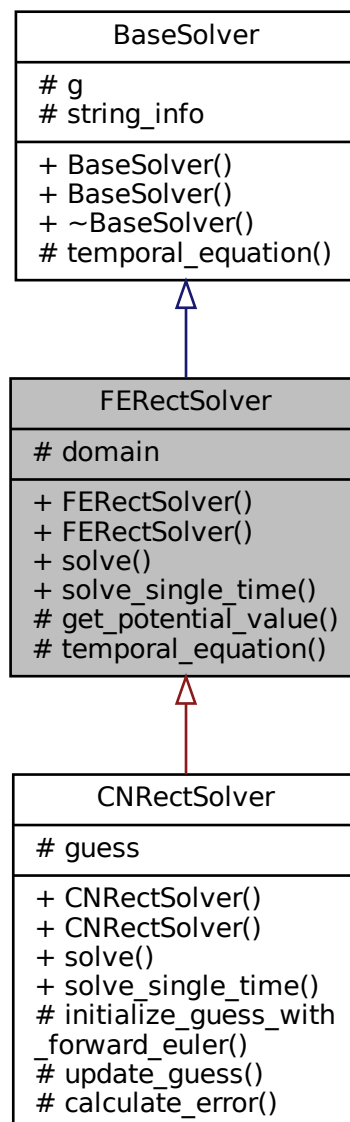
- /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/configuration/[parameters.h](#)

4.10 FERectSolver Class Reference

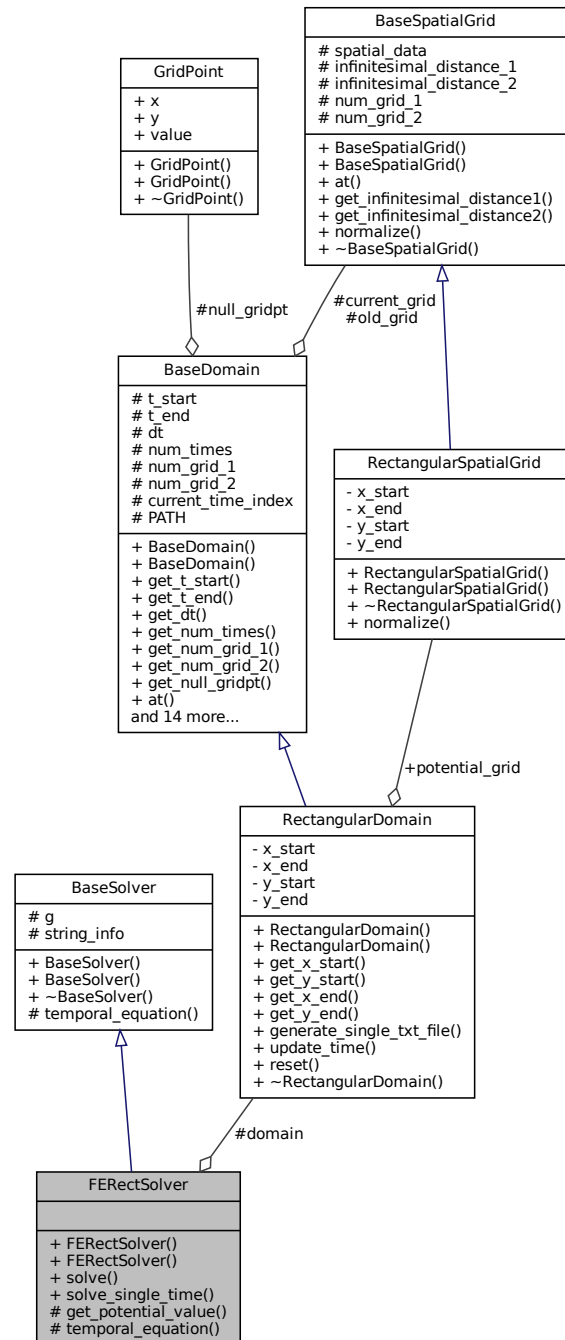
Forward Euler Serial Solver.

```
#include <fe_rect_solver.h>
```

Inheritance diagram for FERectSolver:



Collaboration diagram for FERectSolver:



Public Member Functions

- [FERectSolver](#) ()=default
- [FERectSolver](#) (float [g](#), [RectangularDomain](#) *[domain](#))
Construct a new [FERectSolver::FERectSolver](#) object.
- void [solve](#) (std::string [dir_name](#)="", bool [print_info](#)=true, bool [save_data](#)=true)
Solve Equation.

- void [solve_single_time](#) (int k)
Update $k+1$ th wave function value with k th values.

Protected Member Functions

- float [get_potential_value](#) (int i, int j)
Get Potential value on each grid point.
- std::complex< float > [temporal_equation](#) (int i, int j, int k)
Time differential of phi.

Protected Attributes

- [RectangularDomain](#) * [domain](#)
- float [g](#)
- std::string [string_info](#)

4.10.1 Detailed Description

Forward Euler Serial Solver.

Definition at line 27 of file `fe_rect_solver.h`.

4.10.2 Constructor & Destructor Documentation

4.10.2.1 FERectSolver() [1/2]

```
FERectSolver::FERectSolver ( ) [default]
```

4.10.2.2 FERectSolver() [2/2]

```
FERectSolver::FERectSolver (
    float g_,
    RectangularDomain * domain_ )
```

Construct a new [FERectSolver::FERectSolver](#) object.

Parameters

<i>g_</i>	
<i>domain_</i> ↔	
—	

Definition at line 20 of file fe_rect_solver.cpp.

```

23     : BaseSolver(g_)
24 {
25     this->domain = domain_;
26     this->string_info = std::string{"Forward_Euler_serial_"};
27
28 };

```

4.10.3 Member Function Documentation

4.10.3.1 get_potential_value()

```

float FERectSolver::get_potential_value (
    int i,
    int j ) [protected]

```

Get Potential value on each grid point.

Parameters

<i>i</i>	
<i>j</i>	

Returns

float

Definition at line 37 of file fe_rect_solver.cpp.

```

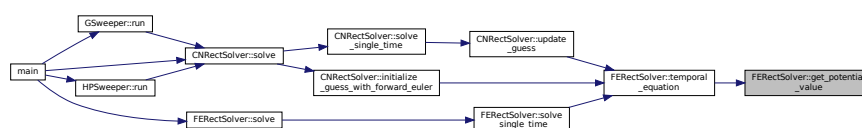
38 {
39
40     return this->domain->potential_grid->at(i, j)->value.real();
41 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



4.10.3.2 solve()

```
void FERectSolver::solve (
    std::string dir_name = "",
    bool print_info = true,
    bool save_data = true )
```

Solve Equation.

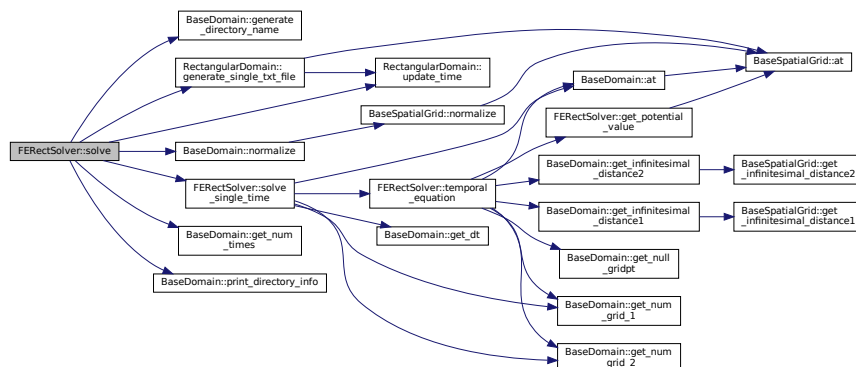
Parameters

<i>dir_name</i>	
<i>print_info</i>	
<i>save_data</i>	

Definition at line 116 of file fe_rect_solver.cpp.

```
117 {
118     int time_length = this->domain->get_num_times();
119     if (save_data) {
120         this->domain->generate_directory_name(this->string_info+dir_name, print_info);
121         //Save initial condition
122         this->domain->generate_single_txt_file(std::string("Solution_") + std::to_string(0));
123     } else {
124         this->domain->update_time();
125     }
126
127     for (int k = 0; k < time_length - 1; ++k)
128     {
129         this->solve_single_time(k);
130         this->domain->normalize(k + 1);
131         if (save_data) {
132             this->domain->generate_single_txt_file(std::string("Solution_") + std::to_string(k + 1));
133         }
134         else {
135             this->domain->update_time();
136         }
137     }
138     if (print_info) {
139         this->domain->print_directory_info();
140     }
141 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.10.3.3 solve_single_time()

```
void FERectSolver::solve_single_time (
    int k )
```

Update k+1 th wave function value with k th values.

Parameters

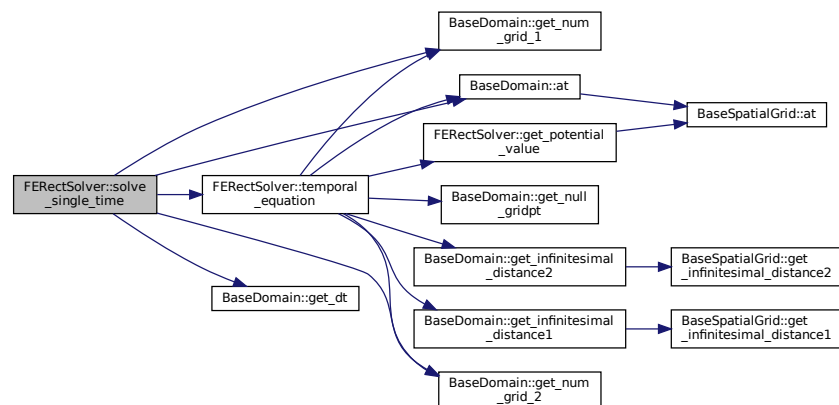
<i>k</i>	
----------	--

Definition at line 95 of file fe_rect_solver.cpp.

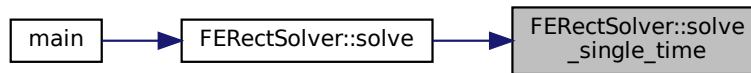
```

96 {
97     int Nx = this->domain->get_num_grid_1();
98     int Ny = this->domain->get_num_grid_2();
99     float dt = this->domain->get_dt();
100     for (int i = 0; i < Nx; ++i)
101     {
102         for (int j = 0; j < Ny; ++j)
103         {
104             (this->domain->at(i, j, k + 1)->value) = this->domain->at(i, j, k)->value + dt *
this->temporal_equation(i, j, k);
105         }
106     }
107 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.10.3.4 temporal_equation()

```
std::complex< float > FRectSolver::temporal_equation (
    int i,
    int j,
    int k ) [protected]
```

Time differential of phi.

Parameters

<i>i</i>	index for x
<i>j</i>	index for y
<i>k</i>	index for time(t)

Returns

std::complex<float> time differential at x, y, t

Definition at line 51 of file fe_rect_solver.cpp.

```

52 {
53     //Use five stencil method
54     auto point_data = this->domain->at(i, j, k);
55
56     //l,r,d,u denotes left, right, down, up value
57     //Check boundary
58     auto point_data_l = this->domain->at(i - 1, j, k);
59     if (i <= 0)
60         point_data_l = this->domain->get_null_gridpt();
61     auto point_data_d = this->domain->at(i, j - 1, k);
62     if (j <= 0)
63         point_data_d = this->domain->get_null_gridpt();
64     auto point_data_r = this->domain->at(i + 1, j, k);
65     if (i >= (this->domain->get_num_grid_1() - 1))
66         point_data_r = this->domain->get_null_gridpt();
67     auto point_data_u = this->domain->at(i, j + 1, k);
68     if (j >= (this->domain->get_num_grid_2() - 1))
69         point_data_u = this->domain->get_null_gridpt();
70
71     //potential at x, y
72     float V_ij = this->get_potential_value(i, j);
73     //this->potential_func(point_data->x, point_data->y);
74
75     //g * |psi(x,y)|^2
76     float additional_term = (this->g) * (std::abs(point_data->value)) * (std::abs(point_data->value));
77
78     //Set infinitesimal value
79     float dx = this->domain->get_infinitesimal_distance1();
80     float dy = this->domain->get_infinitesimal_distance2();
```

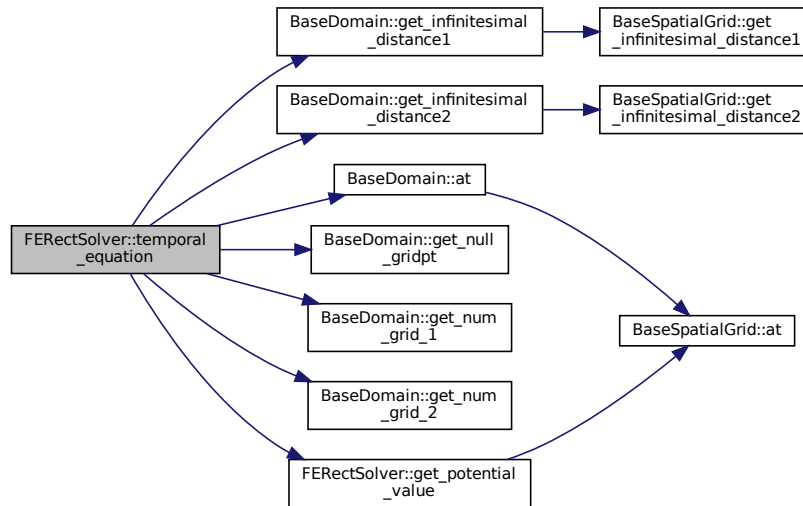


```

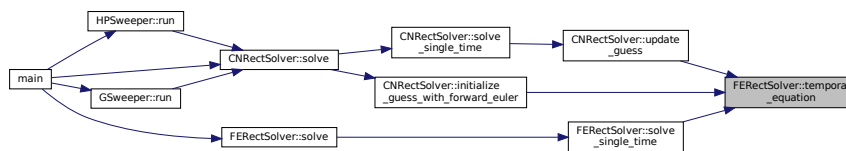
81 //df denote time differential of dt (d(psi)/dt)
82 // = (laplace - V-g|psi|^2) psi
83 std::complex<float> df =
84   +((point_data_r->value) + (point_data_l->value) - (point_data->value) * std::complex<float>{2}) /
85   (std::complex<float>{dx * dx}) + ((point_data_u->value) + (point_data_d->value) - (point_data->value)
86   * std::complex<float>{2}) / (std::complex<float>{dy * dy}) - (V_ij + additional_term) *
87   (point_data->value);
88 df *= std::complex<float>{0, 1};
89
90 return df;
91 };

```

Here is the call graph for this function:



Here is the caller graph for this function:



4.10.4 Member Data Documentation

4.10.4.1 domain

`RectangularDomain*` FERectSolver::domain [protected]

Definition at line 39 of file `fe_rect_solver.h`.

4.10.4.2 g

```
float BaseSolver::g [protected], [inherited]
```

Definition at line 30 of file base_solver.h.

4.10.4.3 string_info

```
std::string BaseSolver::string_info [protected], [inherited]
```

Definition at line 32 of file base_solver.h.

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

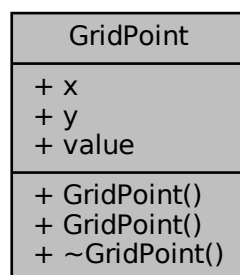
- [/home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/solver/serial_solver/forward_euler/fe_rect_solver.h](#)
- [/home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/solver/serial_solver/forward_euler/fe_rect_solver.cpp](#)

4.11 GridPoint Class Reference

Grid point class (single point)

```
#include <base_domain.h>
```

Collaboration diagram for GridPoint:



Public Member Functions

- [GridPoint](#) ()=default
- [GridPoint](#) (float [x](#), float [y](#), std::complex< float > wave_function)
- [~GridPoint](#) ()

Public Attributes

- float `x`
- float `y`
- `std::complex< float >` `value`

4.11.1 Detailed Description

Grid point class (single point)

Definition at line 31 of file `base_domain.h`.

4.11.2 Constructor & Destructor Documentation

4.11.2.1 GridPoint() [1/2]

```
GridPoint::GridPoint ( ) [default]
```

4.11.2.2 GridPoint() [2/2]

```
GridPoint::GridPoint (
    float x,
    float y,
    std::complex< float > wave_function )
```

Definition at line 13 of file `base_domain.cpp`.

```
13 : x(x_), y(y_), value(wave_function) {}
```

4.11.2.3 ~GridPoint()

```
GridPoint::~GridPoint ( )
```

Definition at line 14 of file `base_domain.cpp`.

```
14 {};
```

4.11.3 Member Data Documentation

4.11.3.1 value

```
std::complex<float> GridPoint::value
```

Definition at line 38 of file `base_domain.h`.

4.11.3.2 x

```
float GridPoint::x
```

Definition at line 36 of file `base_domain.h`.

4.11.3.3 y

```
float GridPoint::y
```

Definition at line 36 of file `base_domain.h`.

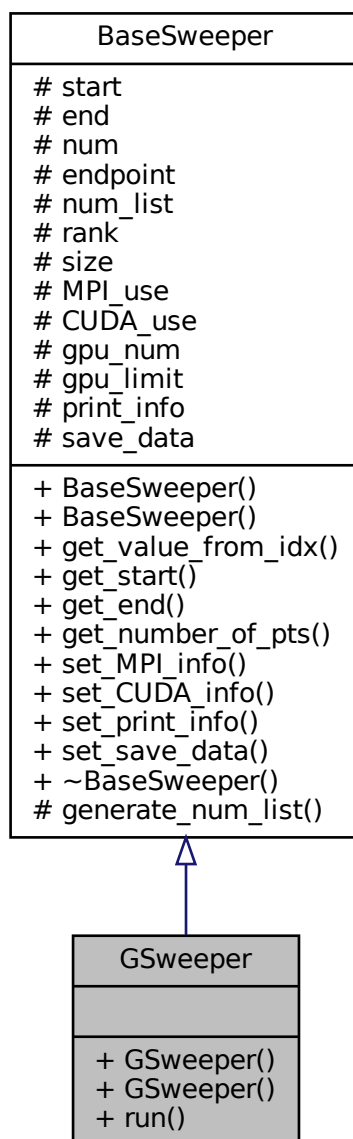
The documentation for this class was generated from the following files:

- `/home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/domain/base_domain.h`
- `/home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/domain/base_domain.cpp`

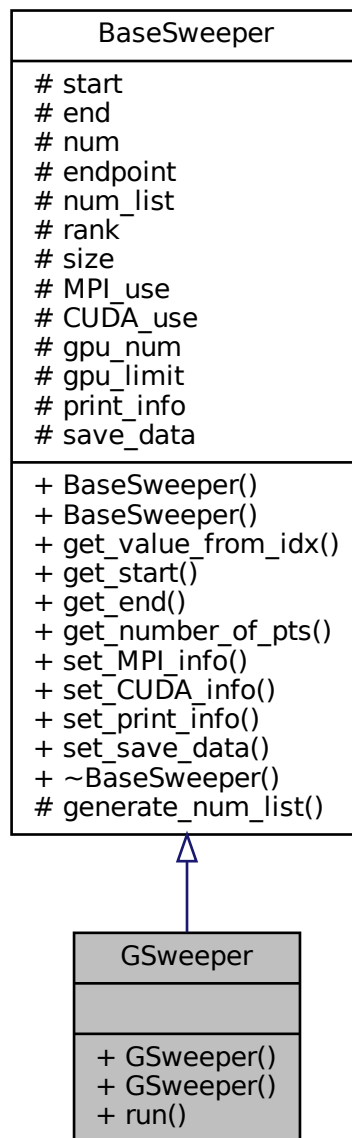
4.12 GSweeper Class Reference

```
#include <g_sweeper.h>
```

Inheritance diagram for GSweeper:



Collaboration diagram for GSweeper:



Public Member Functions

- `GSweeper()` = default
- `GSweeper` (float `start`, float `end`, int `num`, bool `endpoint`)
- void `run` (`RectangularDomain` *domain, `InitialCondition` *initial_condition, `HarmonicPotential` *potential)
- float `get_value_from_idx` (int idx)
- float `get_start` ()
- float `get_end` ()
- int `get_number_of_pts` ()
- virtual void `set_MPI_info` (int rank, int size)

- virtual void `set_CUDA_info` (int `gpu_num`, int `gpu_limit`)
- void `set_print_info` (bool `print_info`)
- void `set_save_data` (bool `save_data`)

Protected Member Functions

- void `generate_num_list` ()
generate number of points that sweep from start to end

Protected Attributes

- float `start`
- float `end`
- int `num`
- bool `endpoint`
- vector< float > `num_list`
- int `rank` =0
- int `size` =0
- bool `MPI_use`
- bool `CUDA_use`
- int `gpu_num` =1
- int `gpu_limit` =3
- bool `print_info` =true
- bool `save_data` =true

4.12.1 Detailed Description

Definition at line 5 of file `g_sweeper.h`.

4.12.2 Constructor & Destructor Documentation

4.12.2.1 GSweeper() [1/2]

```
GSweeper::GSweeper ( ) [default]
```

4.12.2.2 GSweeper() [2/2]

```
GSweeper::GSweeper (
    float start,
    float end,
    int num,
    bool endpoint )
```

Definition at line 7 of file `g_sweeper.cpp`.

```
8 : BaseSweeper(start, end, num, endpoint){
9
10 }
```

4.12.3 Member Function Documentation

4.12.3.1 generate_num_list()

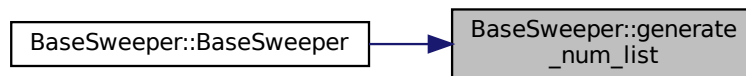
```
void BaseSweeper::generate_num_list ( ) [protected], [inherited]
```

generate number of points that sweep from start to end

Definition at line 15 of file base_sweeper.cpp.

```
15         {
16     this->num_list = std::vector<float>(this->num);
17     float d = 0.;
18     if(this->endpoint){
19         d = (this -> end - this-> start ) / float(this-> num -1 );
20     }else{
21         d = (this -> end - this-> start ) / float(this-> num);
22     }
23
24     for (int i=0; i<this -> num; ++i){
25         this->num_list[i] = this ->start + d*i;
26     }
27
28 }
```

Here is the caller graph for this function:



4.12.3.2 get_end()

```
float BaseSweeper::get_end ( ) [inherited]
```

Definition at line 38 of file base_sweeper.cpp.

```
38     {
39     return end;
40 }
```

4.12.3.3 get_number_of_pts()

```
int BaseSweeper::get_number_of_pts ( ) [inherited]
```

Definition at line 41 of file base_sweeper.cpp.

```
41     {
42     return num;
43 }
```


4.12.3.4 get_start()

```
float BaseSweeper::get_start ( ) [inherited]
```

Definition at line 35 of file base_sweeper.cpp.

```
35     {
36     return start;
37 }
```

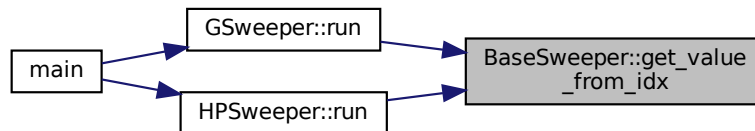
4.12.3.5 get_value_from_idx()

```
float BaseSweeper::get_value_from_idx (
    int idx ) [inherited]
```

Definition at line 31 of file base_sweeper.cpp.

```
31     {
32     return this->num_list[idx];
33 }
```

Here is the caller graph for this function:



4.12.3.6 run()

```
void GSweeper::run (
    RectangularDomain * domain,
    InitialCondition * initial_condition,
    HarmonicPotential * potential )
```

Definition at line 12 of file g_sweeper.cpp.

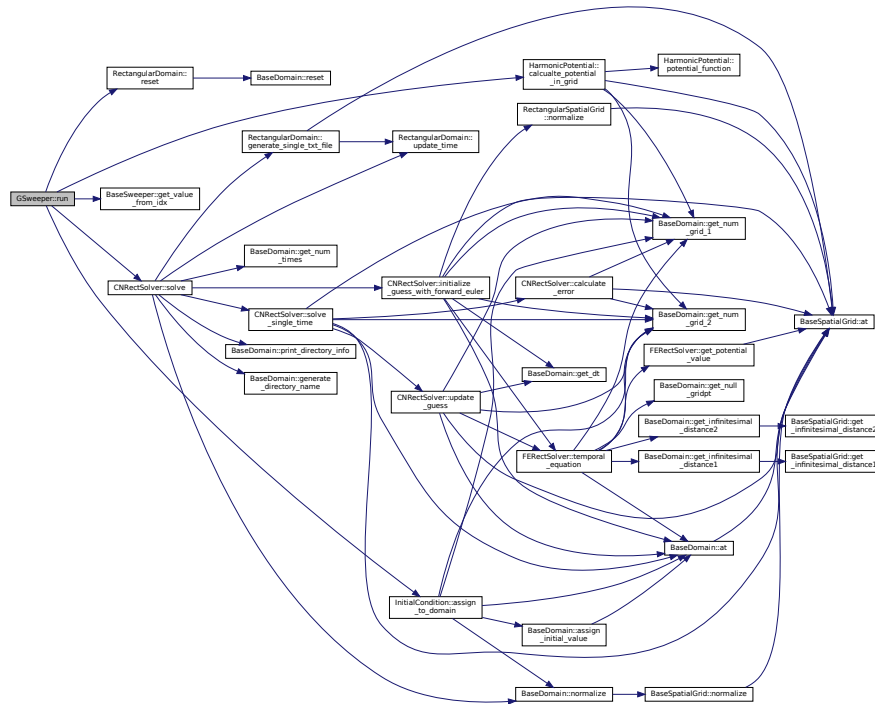
```
12     {
13
14
15     if (!(this -> MPI_use) && !(this -> CUDA_use)) {
16         if (print_info) {
17             cout<< "Running serially started"<<endl;
18         }
19         float g=0;
20         for(int i=0; i<this -> num ; ++i){
21             //Set conditions
22             initial_condition->assign_to_domain(domain);
23             potential->calcualte_potential_in_grid(domain);
24             g = this -> get_value_from_idx(i);
25             //Apply on solver
26             CNRectSolver* solver =new CNRectSolver(g, domain);
27             //solve using solver. It automatically save data
```

```

28         solver->solve(1e-11, 101, to_string(i) , this-> print_info, this -> save_data);
29         //reset domain to use in next iteration
30         domain->reset();
31         delete solver;
32     }
33 }
34 else if ((this->MPI_use) && !(this->CUDA_use)){
35     //If number of tasks exceed number of processors, abort.
36     if(print_info){
37         cout<< "Running with only MPI started"<<endl;
38     }
39     if (num > size){
40         MPI_Abort( MPI_COMM_WORLD, EXIT_FAILURE);
41     }
42     float g = 0 ;
43     if (rank < num){
44         initial_condition->assign_to_domain(domain);
45         potential->calcualte_potential_in_grid(domain);
46         g = this -> get_value_from_idx(rank);
47         CNRectSolver* solver =new CNRectSolver(g, domain);
48         solver->solve(1e-11, 101, "MPI_"+to_string(rank), this -> print_info, this->save_data);
49         delete solver;
50     }else{
51         // No job for extra processors
52         ;
53     }
54 }else if (!(this->MPI_use) && (this->CUDA_use)){
55
56     if(print_info){
57         cout<< "Running with CUDA serially started"<<endl;
58     }
59     float g=0;
60     for(int i=0; i<this -> num ; ++i){
61         //Set conditions
62         initial_condition->assign_to_domain(domain);
63         potential->calcualte_potential_in_grid(domain);
64         g = this -> get_value_from_idx(i);
65         //Apply on solver
66         CNRectPSolver* solver =new CNRectPSolver(g, domain, 0);//solve using solver. It automatically
        save data
67         solver->solve(1e-11, 101, "CUDA_"+to_string(i) , this-> print_info, this -> save_data);
68         //reset domain to use in next iteration
69         domain->reset();
70         delete solver;
71     }
72 }
73 }
74 else{
75     if(print_info){
76         cout<< "Running with CUDA & MPI started"<<endl;
77     }
78     float g=0;
79
80     if (num > size){
81         MPI_Abort( MPI_COMM_WORLD, EXIT_FAILURE);
82     }
83     int max_pallel_tasks = this -> gpu_num * this ->gpu_limit;
84     int repeat = num / max_pallel_tasks+(num%max_pallel_tasks !=0);
85     for(int i=0; i<repeat; ++i){
86         if (i*repeat < max_pallel_tasks && rank < (i+1)*max_pallel_tasks && rank < num ){
87             int casted_GPU_num = (rank % max_pallel_tasks) % gpu_limit;
88             initial_condition->assign_to_domain(domain);
89             potential->calcualte_potential_in_grid(domain);
90             g = this -> get_value_from_idx(rank);
91             CNRectPSolver* solver =new CNRectPSolver(g, domain, casted_GPU_num);
92             solver->solve(1e-11, 101, "MPI&CUDA_"+to_string(rank), this -> print_info,
this->save_data);
93             delete solver;
94         }
95     }
96 }
97
98 }
99 }
100 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



4.12.3.7 set_CUDA_info()

```
void BaseSweeper::set_CUDA_info (
    int gpu_num,
    int gpu_limit ) [virtual], [inherited]
```

Definition at line 51 of file `base_sweeper.cpp`.

```
51 {
52     this -> CUDA_use=true;
53     this -> gpu_num = gpu_num;
54     this -> gpu_limit = gpu_limit;
55 }
```

Here is the caller graph for this function:



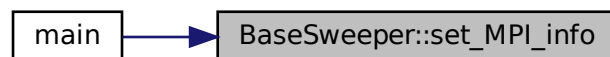
4.12.3.8 set_MPI_info()

```
void BaseSweeper::set_MPI_info (
    int rank,
    int size ) [virtual], [inherited]
```

Definition at line 45 of file `base_sweeper.cpp`.

```
45     {
46         this -> rank = rank;
47         this -> size = size;
48         this -> MPI_use=true;
49     }
```

Here is the caller graph for this function:



4.12.3.9 set_print_info()

```
void BaseSweeper::set_print_info (
    bool print_info ) [inherited]
```

Definition at line 57 of file `base_sweeper.cpp`.

```
57     {
58         this -> print_info = print_info;
59     }
```

Here is the caller graph for this function:



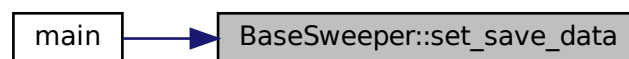
4.12.3.10 set_save_data()

```
void BaseSweeper::set_save_data (
    bool save_data ) [inherited]
```

Definition at line 60 of file `base_sweeper.cpp`.

```
60 {
61     this -> save_data = save_data;
62 }
```

Here is the caller graph for this function:



4.12.4 Member Data Documentation

4.12.4.1 CUDA_use

```
bool BaseSweeper::CUDA_use [protected], [inherited]
```

Definition at line 36 of file `base_sweeper.h`.

4.12.4.2 end

```
float BaseSweeper::end [protected], [inherited]
```

Definition at line 25 of file base_sweeper.h.

4.12.4.3 endpoint

```
bool BaseSweeper::endpoint [protected], [inherited]
```

Definition at line 27 of file base_sweeper.h.

4.12.4.4 gpu_limit

```
int BaseSweeper::gpu_limit =3 [protected], [inherited]
```

Definition at line 38 of file base_sweeper.h.

4.12.4.5 gpu_num

```
int BaseSweeper::gpu_num =1 [protected], [inherited]
```

Definition at line 37 of file base_sweeper.h.

4.12.4.6 MPI_use

```
bool BaseSweeper::MPI_use [protected], [inherited]
```

Definition at line 34 of file base_sweeper.h.

4.12.4.7 num

```
int BaseSweeper::num [protected], [inherited]
```

Definition at line 26 of file base_sweeper.h.

4.12.4.8 num_list

```
vector<float> BaseSweeper::num_list [protected], [inherited]
```

Definition at line 28 of file base_sweeper.h.

4.12.4.9 print_info

```
bool BaseSweeper::print_info =true [protected], [inherited]
```

Definition at line 40 of file base_sweeper.h.

4.12.4.10 rank

```
int BaseSweeper::rank =0 [protected], [inherited]
```

Definition at line 32 of file base_sweeper.h.

4.12.4.11 save_data

```
bool BaseSweeper::save_data =true [protected], [inherited]
```

Definition at line 41 of file base_sweeper.h.

4.12.4.12 size

```
int BaseSweeper::size =0 [protected], [inherited]
```

Definition at line 33 of file base_sweeper.h.

4.12.4.13 start

```
float BaseSweeper::start [protected], [inherited]
```

Definition at line 24 of file base_sweeper.h.

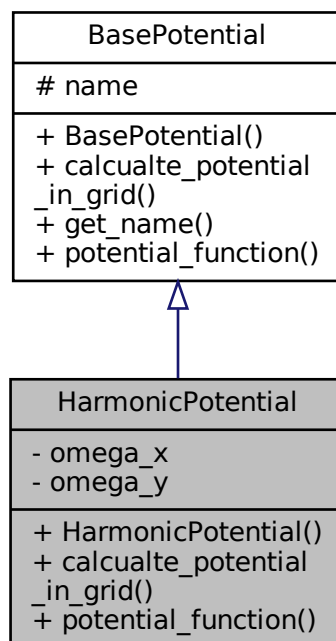
The documentation for this class was generated from the following files:

- [/home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/sweeper/g_sweeper.h](#)
- [/home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/sweeper/g_sweeper.cpp](#)

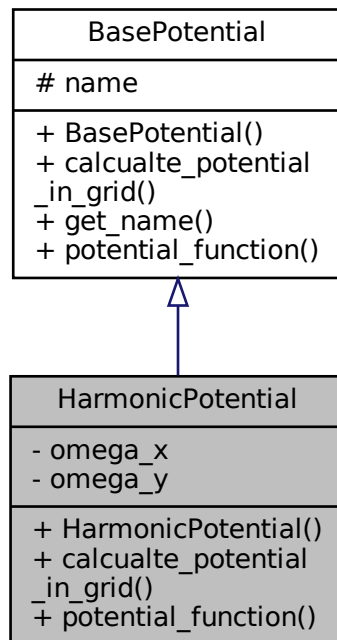
4.13 HarmonicPotential Class Reference

```
#include <harmonic_potential.h>
```

Inheritance diagram for HarmonicPotential:



Collaboration diagram for HarmonicPotential:



Public Member Functions

- [HarmonicPotential](#) (float [omega_x](#), float [omega_y](#))
Construct a new Harmonic Potential:: Harmonic Potential object $V = 0.5 * (\text{omega_x}^2 * x^2 + \text{omega_y}^2 * y^2)$
- void [calcualte_potential_in_grid](#) ([RectangularDomain](#) *domain)
- float [potential_function](#) (float x, float y)
- std::string [get_name](#) ()
Getter for name class variable.

Protected Attributes

- std::string [name](#)

Private Attributes

- float [omega_x](#)
- float [omega_y](#)

4.13.1 Detailed Description

Definition at line 19 of file harmonic_potential.h.

4.13.2 Constructor & Destructor Documentation

4.13.2.1 HarmonicPotential()

```
HarmonicPotential::HarmonicPotential (
    float omega_x,
    float omega_y )
```

Construct a new Harmonic Potential:: Harmonic Potential object $V = 0.5 * (\omega_x^2 * x^2 + \omega_y^2 * y^2)$

Parameters

<i>omega_x</i>	
<i>omega_y</i>	

Definition at line 19 of file harmonic_potential.cpp.

```
20 : BasePotential(), omega_x(omega_x), omega_y(omega_y)
21 {
22     this->name = std::string("Harmonic");
23 }
```

4.13.3 Member Function Documentation

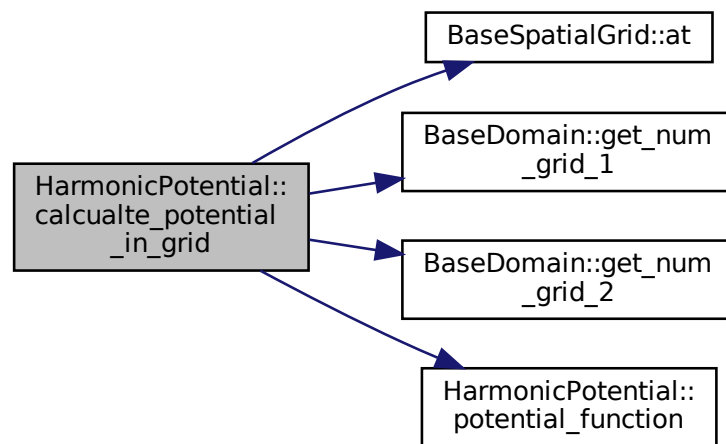
4.13.3.1 calculalte_potential_in_grid()

```
void HarmonicPotential::calculalte_potential_in_grid (
    RectangularDomain * domain )
```

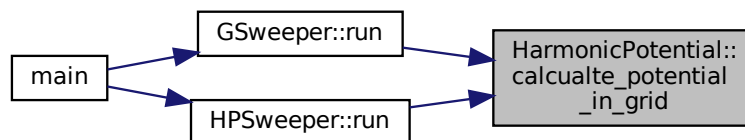
Definition at line 29 of file harmonic_potential.cpp.

```
30 {
31     int num_grid_1 = domain->get_num_grid_1();
32     int num_grid_2 = domain->get_num_grid_2();
33     for (auto i = 0; i < num_grid_1; ++i)
34     {
35         for (auto j = 0; j < num_grid_2; ++j)
36         {
37             auto point = domain->potential_grid->at(i, j);
38
39             //Assign potential value in potential grid
40             point->value = std::complex<float>{this->potential_function(point->x, point->y), 0};
41         }
42     }
43 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.13.3.2 get_name()

```
std::string BasePotential::get_name ( ) [inherited]
```

Getter for name class variable.

Returns

`std::string`

Definition at line 40 of file `base_potential.cpp`.

```

41 {
42     return this->name;
43 }
```

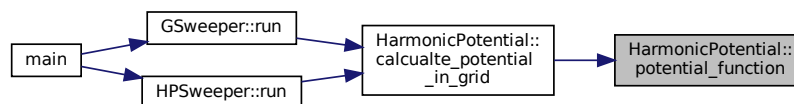
4.13.3.3 potential_function()

```
float HarmonicPotential::potential_function (
    float x,
    float y )
```

Definition at line 25 of file harmonic_potential.cpp.

```
26 {
27     return 0.5 * (this->omega_x * x * x + this->omega_y * y * y);
28 }
```

Here is the caller graph for this function:



4.13.4 Member Data Documentation

4.13.4.1 name

```
std::string BasePotential::name [protected], [inherited]
```

Definition at line 27 of file base_potential.h.

4.13.4.2 omega_x

```
float HarmonicPotential::omega_x [private]
```

Definition at line 27 of file harmonic_potential.h.

4.13.4.3 omega_y

```
float HarmonicPotential::omega_y [private]
```

Definition at line 27 of file harmonic_potential.h.

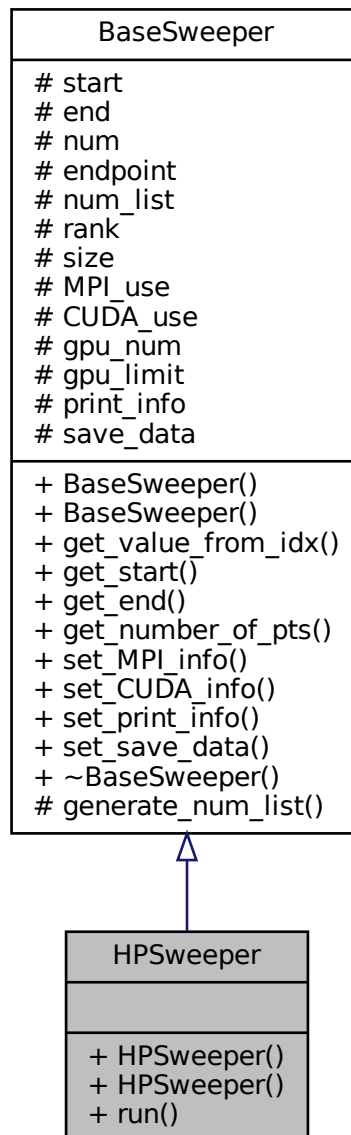
The documentation for this class was generated from the following files:

- [/home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/potential/harmonic_potential.h](#)
- [/home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/potential/harmonic_potential.cpp](#)

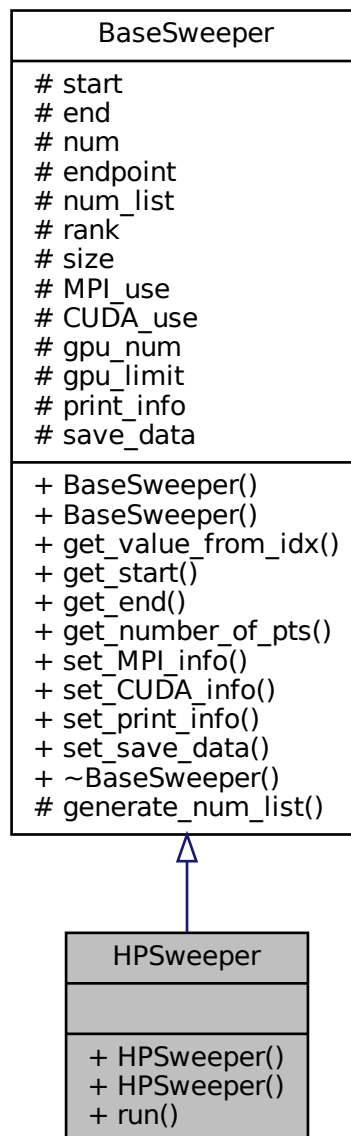
4.14 HPSweeper Class Reference

```
#include <harmonic_p_sweeper.h>
```

Inheritance diagram for HPSweeper:



Collaboration diagram for HPSweeper:



Public Member Functions

- [HPSweeper](#) ()=default
- [HPSweeper](#) (float [start](#), float [end](#), int [num](#), bool [endpoint](#))
- void [run](#) ([RectangularDomain](#) *domain, [InitialCondition](#) *initial_condition, float g)
- float [get_value_from_idx](#) (int idx)
- float [get_start](#) ()
- float [get_end](#) ()
- int [get_number_of_pts](#) ()
- virtual void [set_MPI_info](#) (int [rank](#), int [size](#))

- virtual void `set_CUDA_info` (int `gpu_num`, int `gpu_limit`)
- void `set_print_info` (bool `print_info`)
- void `set_save_data` (bool `save_data`)

Protected Member Functions

- void `generate_num_list` ()
generate number of points that sweep from start to end

Protected Attributes

- float `start`
- float `end`
- int `num`
- bool `endpoint`
- vector< float > `num_list`
- int `rank` =0
- int `size` =0
- bool `MPI_use`
- bool `CUDA_use`
- int `gpu_num` =1
- int `gpu_limit` =3
- bool `print_info` =true
- bool `save_data` =true

4.14.1 Detailed Description

Definition at line 5 of file `harmonic_p_sweeper.h`.

4.14.2 Constructor & Destructor Documentation

4.14.2.1 HPSweeper() [1/2]

```
HPSweeper::HPSweeper ( ) [default]
```

4.14.2.2 HPSweeper() [2/2]

```
HPSweeper::HPSweeper (
    float start,
    float end,
    int num,
    bool endpoint )
```

Definition at line 7 of file `harmonic_p_sweeper.cpp`.

```
8 : BaseSweeper(start, end, num, endpoint){
9     //numlist contains assymetry of angular frequency
10 }
```

4.14.3 Member Function Documentation

4.14.3.1 generate_num_list()

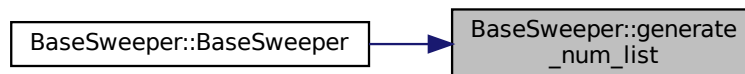
```
void BaseSweeper::generate_num_list ( ) [protected], [inherited]
```

generate number of points that sweep from start to end

Definition at line 15 of file base_sweeper.cpp.

```
15         {
16     this->num_list = std::vector<float>(this->num);
17     float d = 0.;
18     if(this->endpoint){
19         d = (this -> end - this-> start ) / float(this-> num -1 );
20     }else{
21         d = (this -> end - this-> start ) / float(this-> num);
22     }
23
24     for (int i=0; i<this -> num; ++i){
25         this->num_list[i] = this ->start + d*i;
26     }
27
28 }
```

Here is the caller graph for this function:



4.14.3.2 get_end()

```
float BaseSweeper::get_end ( ) [inherited]
```

Definition at line 38 of file base_sweeper.cpp.

```
38     {
39     return end;
40 }
```

4.14.3.3 get_number_of_pts()

```
int BaseSweeper::get_number_of_pts ( ) [inherited]
```

Definition at line 41 of file base_sweeper.cpp.

```
41     {
42     return num;
43 }
```


4.14.3.4 get_start()

```
float BaseSweeper::get_start ( ) [inherited]
```

Definition at line 35 of file base_sweeper.cpp.

```
35     {
36     return start;
37 }
```

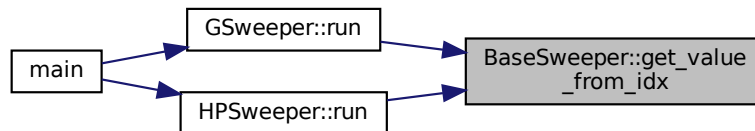
4.14.3.5 get_value_from_idx()

```
float BaseSweeper::get_value_from_idx (
    int idx ) [inherited]
```

Definition at line 31 of file base_sweeper.cpp.

```
31     {
32     return this->num_list[idx];
33 }
```

Here is the caller graph for this function:



4.14.3.6 run()

```
void HPSweeper::run (
    RectangularDomain * domain,
    InitialCondition * initial_condition,
    float g )
```

Definition at line 12 of file harmonic_p_sweeper.cpp.

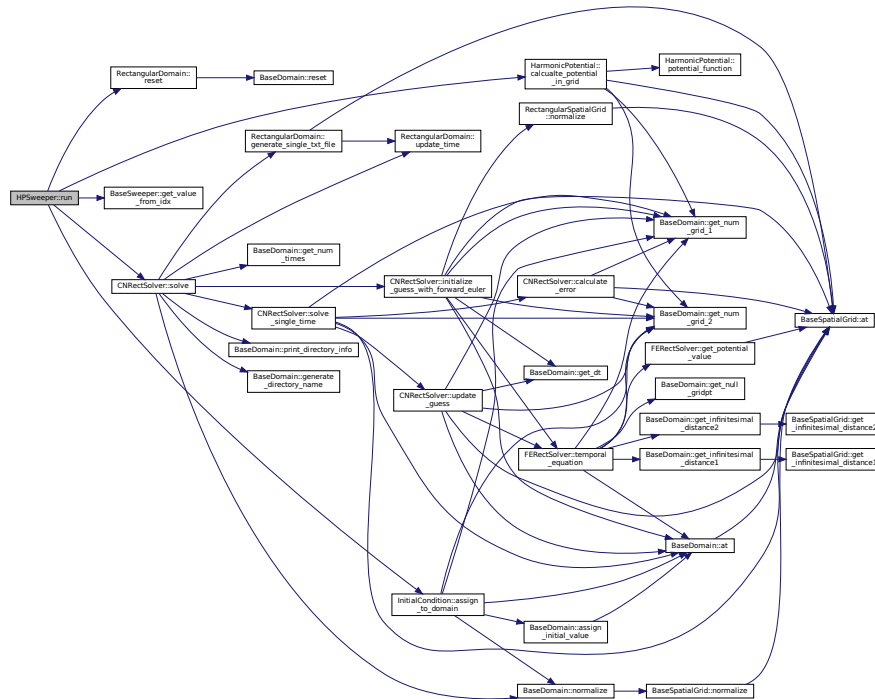
```
12
13
14
15     if (!(this -> MPI_use) && !(this -> CUDA_use)){
16         if(print_info){
17             cout<< "Running serially started" << endl;
18         }
19
20         for(int i=0; i<this -> num ; ++i){
21             //Set conditions
22             initial_condition->assign_to_domain(domain);
23             HarmonicPotential * potential = new HarmonicPotential(1, get_value_from_idx(i));
24             potential->calcualte_potential_in_grid(domain);
25             g = this -> get_value_from_idx(i);
26             //Apply on solver
27             CNRectSolver* solver =new CNRectSolver(g, domain);
28             //solve using solver. It automatically save data
29         }
```

```

29         solver->solve(1e-11, 101, to_string(i) , this-> print_info, this -> save_data);
30         //reset domain to use in next iteration
31         domain->reset();
32         delete solver;
33         delete potential;
34     }
35 }
36 else if ((this->MPI_use) && !(this->CUDA_use)){
37     //If number of tasks exceed number of processors, abort.
38     if(print_info){
39         cout<< "Running with only MPI started"<<endl;
40     }
41     if (num > size){
42         MPI_Abort( MPI_COMM_WORLD, EXIT_FAILURE);
43     }
44     float g = 0 ;
45     if (rank < num){
46         initial_condition->assign_to_domain(domain);
47         HarmonicPotential * potential =new HarmonicPotential(1, get_value_from_idx(rank));
48         potential -> calcualte_potential_in_grid(domain);
49         CNRectSolver* solver =new CNRectSolver(g, domain);
50         solver->solve(1e-11, 101, "MPI_"+to_string(rank), this -> print_info, this->save_data);
51         delete solver;
52         delete potential;
53     }else{
54         // No job for extra processors
55         ;
56     }
57 }else if (!(this->MPI_use) && (this->CUDA_use)){
58
59     if(print_info){
60         cout<< "Running with CUDA serially started"<<endl;
61     }
62     for(int i=0; i<this -> num ; ++i){
63         //Set conditions
64         initial_condition->assign_to_domain(domain);
65         HarmonicPotential * potential = new HarmonicPotential(1, get_value_from_idx(i));
66         potential->calcualte_potential_in_grid(domain);
67         //Apply on solver
68         CNRectPSolver* solver =new CNRectPSolver(g, domain, 0);//solve using solver. It automatically
        save data
69         solver->solve(1e-11, 101, "CUDA_"+to_string(i) , this-> print_info, this -> save_data);
70         //reset domain to use in next iteration
71         domain->reset();
72         delete solver;
73         delete potential;
74     }
75 }
76 }
77 else{
78     if(print_info){
79         cout<< "Running with CUDA & MPI started"<<endl;
80     }
81     float g=0;
82
83     if (num > size){
84         MPI_Abort( MPI_COMM_WORLD, EXIT_FAILURE);
85     }
86     int max_pallel_tasks = this -> gpu_num * this ->gpu_limit;
87     int repeat = num / max_pallel_tasks+(num%max_pallel_tasks !=0);
88     for(int i=0; i<repeat; ++i){
89         if (i*repeat < max_pallel_tasks && rank < (i+1)*max_pallel_tasks && rank < num ){
90             int casted_GPU_num = (rank % max_pallel_tasks) % gpu_limit;
91             initial_condition->assign_to_domain(domain);
92             auto potential = new HarmonicPotential(1, get_value_from_idx(rank));
93             potential->calcualte_potential_in_grid(domain);
94             //g = this -> get_value_from_idx(rank);
95             CNRectPSolver* solver =new CNRectPSolver(g, domain, casted_GPU_num);
96             solver->solve(1e-11, 101, "MPI&CUDA_"+to_string(rank), this -> print_info,
            this->save_data);
97             delete solver;
98             delete potential;
99         }
100     }
101 }
102
103
104 }
105 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



4.14.3.7 set_CUDA_info()

```
void BaseSweeper::set_CUDA_info (
    int gpu_num,
    int gpu_limit ) [virtual], [inherited]
```

Definition at line 51 of file base_sweeper.cpp.

```

51
52     this -> CUDA_use=true;
53     this -> gpu_num = gpu_num;
54     this -> gpu_limit = gpu_limit;
55 }

```

Here is the caller graph for this function:



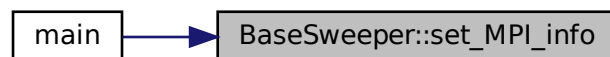
4.14.3.8 set_MPI_info()

```
void BaseSweeper::set_MPI_info (
    int rank,
    int size ) [virtual], [inherited]
```

Definition at line 45 of file `base_sweeper.cpp`.

```
45     {
46         this -> rank = rank;
47         this -> size = size;
48         this -> MPI_use=true;
49     }
```

Here is the caller graph for this function:



4.14.3.9 set_print_info()

```
void BaseSweeper::set_print_info (
    bool print_info ) [inherited]
```

Definition at line 57 of file `base_sweeper.cpp`.

```
57     {
58         this -> print_info = print_info;
59     }
```

Here is the caller graph for this function:



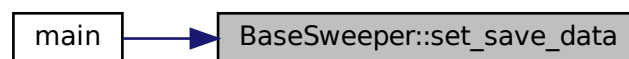
4.14.3.10 set_save_data()

```
void BaseSweeper::set_save_data (
    bool save_data ) [inherited]
```

Definition at line 60 of file `base_sweeper.cpp`.

```
60 {
61     this -> save_data = save_data;
62 }
```

Here is the caller graph for this function:



4.14.4 Member Data Documentation

4.14.4.1 CUDA_use

```
bool BaseSweeper::CUDA_use [protected], [inherited]
```

Definition at line 36 of file `base_sweeper.h`.

4.14.4.2 end

```
float BaseSweeper::end [protected], [inherited]
```

Definition at line 25 of file base_sweeper.h.

4.14.4.3 endpoint

```
bool BaseSweeper::endpoint [protected], [inherited]
```

Definition at line 27 of file base_sweeper.h.

4.14.4.4 gpu_limit

```
int BaseSweeper::gpu_limit =3 [protected], [inherited]
```

Definition at line 38 of file base_sweeper.h.

4.14.4.5 gpu_num

```
int BaseSweeper::gpu_num =1 [protected], [inherited]
```

Definition at line 37 of file base_sweeper.h.

4.14.4.6 MPI_use

```
bool BaseSweeper::MPI_use [protected], [inherited]
```

Definition at line 34 of file base_sweeper.h.

4.14.4.7 num

```
int BaseSweeper::num [protected], [inherited]
```

Definition at line 26 of file base_sweeper.h.

4.14.4.8 num_list

```
vector<float> BaseSweeper::num_list [protected], [inherited]
```

Definition at line 28 of file base_sweeper.h.

4.14.4.9 print_info

```
bool BaseSweeper::print_info =true [protected], [inherited]
```

Definition at line 40 of file base_sweeper.h.

4.14.4.10 rank

```
int BaseSweeper::rank =0 [protected], [inherited]
```

Definition at line 32 of file base_sweeper.h.

4.14.4.11 save_data

```
bool BaseSweeper::save_data =true [protected], [inherited]
```

Definition at line 41 of file base_sweeper.h.

4.14.4.12 size

```
int BaseSweeper::size =0 [protected], [inherited]
```

Definition at line 33 of file base_sweeper.h.

4.14.4.13 start

```
float BaseSweeper::start [protected], [inherited]
```

Definition at line 24 of file base_sweeper.h.

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

- /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/sweeper/[harmonic_p_sweeper.h](#)
- /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/sweeper/[harmonic_p_sweeper.cpp](#)

4.15 InitialCondition Class Reference

Initial condition class.

```
#include <initial_condition.h>
```

Collaboration diagram for InitialCondition:

InitialCondition
- initial_condition_function
+ InitialCondition() + InitialCondition() + assign_to_domain()

Public Member Functions

- [InitialCondition](#) ()=default
- [InitialCondition](#) (std::function< std::complex< float >(float, float)> [initial_condition_function](#))
Construct a new Initial Condition:: Initial Condition object.
- void [assign_to_domain](#) ([RectangularDomain](#) *domain)
Assign initial condition to the domain class instance.

Private Attributes

- std::function< std::complex< float >(float, float)> [initial_condition_function](#)

4.15.1 Detailed Description

Initial condition class.

Definition at line 24 of file initial_condition.h.

4.15.2 Constructor & Destructor Documentation

4.15.2.1 InitialCondition() [1/2]

```
InitialCondition::InitialCondition ( ) [default]
```

4.15.2.2 InitialCondition() [2/2]

```
InitialCondition::InitialCondition (
    std::function< std::complex< float >(float, float)> initial_condition_function )
```

Construct a new Initial Condition:: Initial Condition object.

Parameters

<code>initial_condition_function</code>

Definition at line 18 of file `initial_condition.cpp`.

```
19 {
20     this->initial_condition_function = initial_condition_function;
21 }
```

4.15.3 Member Function Documentation

4.15.3.1 assign_to_domain()

```
void InitialCondition::assign_to_domain (
    RectangularDomain * domain )
```

Assign initial condition to the domain class instance.

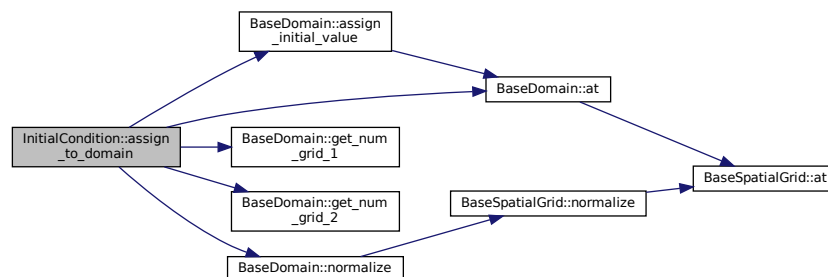
Parameters

<code>domain</code>

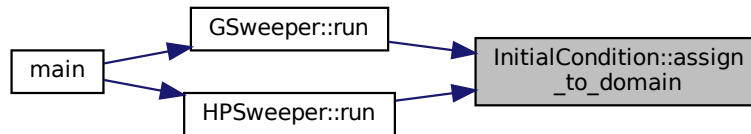
Definition at line 28 of file `initial_condition.cpp`.

```
29 {
30     for (auto i = 0; i < domain->get_num_grid_1(); ++i)
31     {
32         for (auto j = 0; j < domain->get_num_grid_2(); ++j)
33         {
34             auto point_data = domain->at(i, j, 0);
35             domain->assign_initial_value(i, j, this->initial_condition_function(point_data->x,
36             point_data->y));
37         }
38     }
39     domain->normalize(0);
40 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.15.4 Member Data Documentation

4.15.4.1 initial_condition_function

```
std::function<std::complex<float>float, float>> InitialCondition::initial_condition_function
[private]
```

Definition at line 27 of file `initial_condition.h`.

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

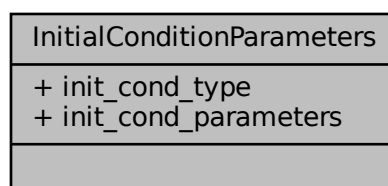
- /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/initial_condition/initial_condition.h
- /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/initial_condition/initial_condition.cpp

4.16 InitialConditionParameters Class Reference

Initial condition branch parameters containing information about initial condition.

```
#include <parameters.h>
```

Collaboration diagram for `InitialConditionParameters`:



Public Attributes

- `std::string` [init_cond_type](#) = `std::string("")`
- `std::map< std::string, float >` [init_cond_parameters](#)

4.16.1 Detailed Description

Initial condition branch parameters containing information about initial condition.

Definition at line 46 of file `parameters.h`.

4.16.2 Member Data Documentation

4.16.2.1 `init_cond_parameters`

```
std::map<std::string, float> InitialConditionParameters::init_cond_parameters
```

Definition at line 50 of file `parameters.h`.

4.16.2.2 `init_cond_type`

```
std::string InitialConditionParameters::init_cond_type = std::string("")
```

Definition at line 49 of file `parameters.h`.

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

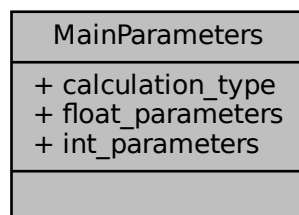
- `/home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/configuration/parameters.h`

4.17 MainParameters Class Reference

Main branch parameters containing information about overall calculation.

```
#include <parameters.h>
```

Collaboration diagram for MainParameters:



Public Attributes

- `std::string` [calculation_type](#) = `std::string("")`
- `std::map< std::string, float >` [float_parameters](#)
- `std::map< std::string, int >` [int_parameters](#)

4.17.1 Detailed Description

Main branch parameters containing information about overall calculation.

Definition at line 21 of file `parameters.h`.

4.17.2 Member Data Documentation

4.17.2.1 `calculation_type`

```
std::string MainParameters::calculation_type = std::string("")
```

Definition at line 24 of file `parameters.h`.

4.17.2.2 `float_parameters`

```
std::map<std::string, float> MainParameters::float_parameters
```

Definition at line 25 of file `parameters.h`.

4.17.2.3 `int_parameters`

```
std::map<std::string, int> MainParameters::int_parameters
```

Definition at line 26 of file `parameters.h`.

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

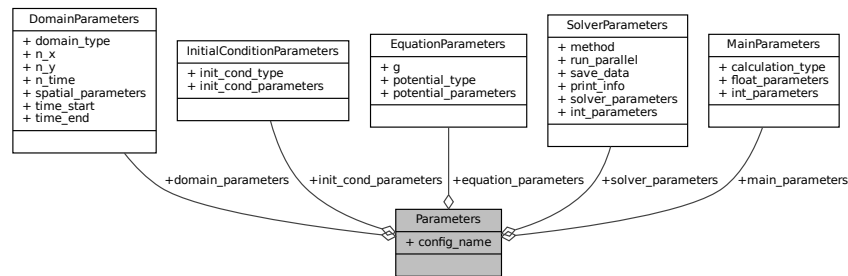
- `/home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/configuration/parameters.h`

4.18 Parameters Class Reference

[Parameters](#) containing configuration name and all other branched parameters.

```
#include <parameters.h>
```

Collaboration diagram for Parameters:



Public Attributes

- `std::string` [config_name](#)
- [MainParameters](#) `main_parameters`
- [DomainParameters](#) `domain_parameters`
- [InitialConditionParameters](#) `init_cond_parameters`
- [EquationParameters](#) `equation_parameters`
- [SolverParameters](#) `solver_parameters`

4.18.1 Detailed Description

[Parameters](#) containing configuration name and all other branched parameters.

Definition at line 84 of file `parameters.h`.

4.18.2 Member Data Documentation

4.18.2.1 config_name

```
std::string Parameters::config_name
```

Definition at line 87 of file `parameters.h`.

4.18.2.2 domain_parameters

`DomainParameters` `Parameters::domain_parameters`

Definition at line 89 of file `parameters.h`.

4.18.2.3 equation_parameters

`EquationParameters` `Parameters::equation_parameters`

Definition at line 91 of file `parameters.h`.

4.18.2.4 init_cond_parameters

`InitialConditionParameters` `Parameters::init_cond_parameters`

Definition at line 90 of file `parameters.h`.

4.18.2.5 main_parameters

`MainParameters` `Parameters::main_parameters`

Definition at line 88 of file `parameters.h`.

4.18.2.6 solver_parameters

`SolverParameters` `Parameters::solver_parameters`

Definition at line 92 of file `parameters.h`.

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

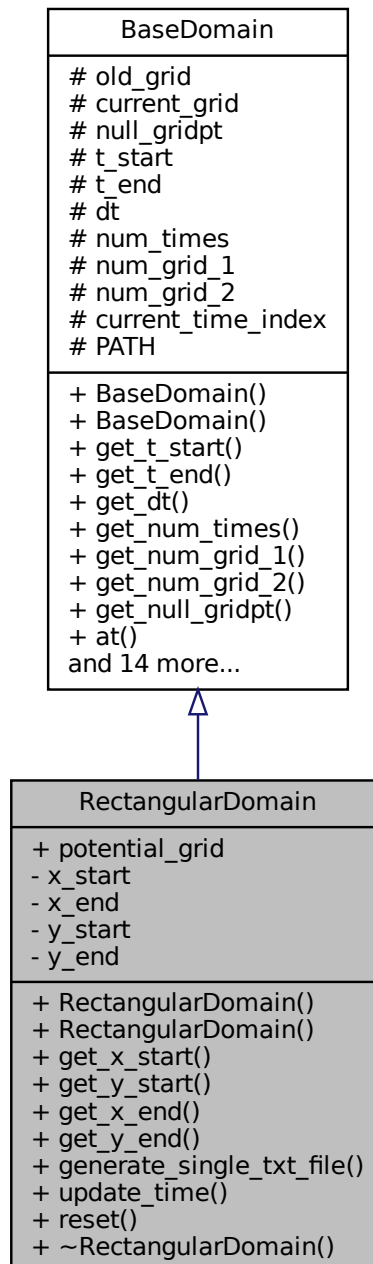
- `/home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/configuration/parameters.h`

4.19 RectangularDomain Class Reference

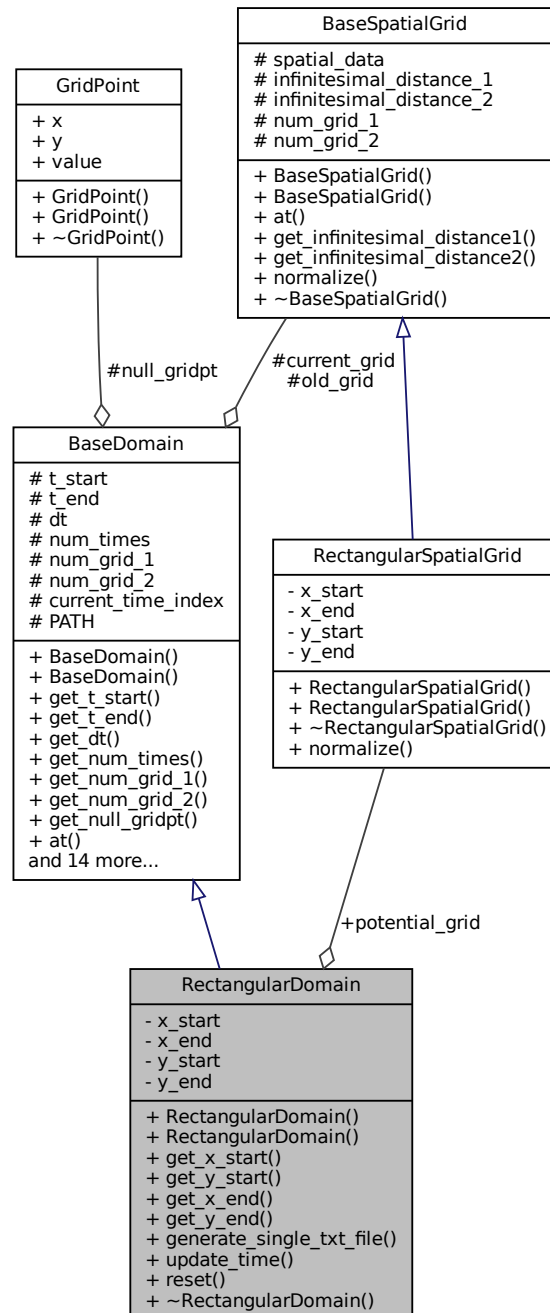
Rectangular domain containing multiple timesteps.

```
#include <rect_domain.h>
```

Inheritance diagram for RectangularDomain:



Collaboration diagram for RectangularDomain:



Public Member Functions

- [RectangularDomain](#) ()=default
- [RectangularDomain](#) (int num_grid_1, int num_grid_2, float t_start, float t_end, int num_times, float x_start, float x_end, float y_start, float y_end)
Construct a new Rectangular Domain:: Rectangular Domain object.
- float [get_x_start](#) ()

- float [get_y_start](#) ()
- float [get_x_end](#) ()
- float [get_y_end](#) ()
- void [generate_single_txt_file](#) (std::string filename, bool cuda_mode=false)
- void [update_time](#) (bool cuda_mode=false)
- void [reset](#) ()
- [~RectangularDomain](#) ()
- float [get_t_start](#) ()
- float [get_t_end](#) ()
- float [get_dt](#) ()
- int [get_num_times](#) ()
- int [get_num_grid_1](#) ()
- int [get_num_grid_2](#) ()
- [GridPoint](#) * [get_null_gridpt](#) ()
- [GridPoint](#) * [at](#) (int index_1, int index_2, int time_index)
- void [assign_initial_value](#) (int index_1, int index_2, std::complex< float > value)
Assign initial value.
- void [assign_wave_function](#) (int index_1, int index_2, int time_index, std::complex< float > value)
- float [time_at](#) (int time_index)
- float [get_infinitesimal_distance1](#) ()
- float [get_infinitesimal_distance2](#) ()
- void [generate_directory_name](#) (std::string info, bool print_info=true)
create directory and return directory name with "/" directory name : "./results/%d-%m-%Y %H-%M-%S_ "+info for exmaple, "./results/"
- void [normalize](#) (int time_index)
- void [print_directory_info](#) ()
- int [get_current_time_index](#) ()
- std::string [get_path](#) ()

Public Attributes

- [RectangularSpatialGrid](#) * [potential_grid](#)

Protected Attributes

- [BaseSpatialGrid](#) * [old_grid](#)
- [BaseSpatialGrid](#) * [current_grid](#)
- [GridPoint](#) * [null_gridpt](#)
- float [t_start](#)
- float [t_end](#)
- float [dt](#)
- int [num_times](#)
- int [num_grid_1](#)
- int [num_grid_2](#)
- int [current_time_index](#) = 0
- std::string [PATH](#)

Private Attributes

- float [x_start](#)
- float [x_end](#)
- float [y_start](#)
- float [y_end](#)

4.19.1 Detailed Description

Rectangular domain containing multiple timesteps.

Definition at line 37 of file rect_domain.h.

4.19.2 Constructor & Destructor Documentation

4.19.2.1 RectangularDomain() [1/2]

```
RectangularDomain::RectangularDomain ( ) [default]
```

4.19.2.2 RectangularDomain() [2/2]

```
RectangularDomain::RectangularDomain (
    int num_grid_1,
    int num_grid_2,
    float t_start,
    float t_end,
    int num_times,
    float x_start,
    float x_end,
    float y_start,
    float y_end )
```

Construct a new Rectangular Domain:: Rectangular Domain object.

Parameters

<i>num_grid_1</i>	Number of grids in x axis
<i>num_grid_2</i>	Number of grids in y axis
<i>t_start</i>	Initial time
<i>t_end</i>	Final time
<i>num_times</i>	iteration number or number of time points
<i>x_start</i>	Start point of x axis
<i>x_end</i>	End point of x axis
<i>y_start</i>	Start point of y axis
<i>y_end</i>	Start point of x axis

Definition at line 92 of file rect_domain.cpp.

```
103 : BaseDomain(num_grid_1, num_grid_2, t_start, t_end, num_times),
104   x_start(x_start),
105   x_end(x_end),
106   y_start(y_start),
```

```

107         y_end(y_end)
108 {
109
110     delete (this->old_grid);
111     delete (this->current_grid);
112     this->old_grid = new RectangularSpatialGrid(num_grid_1, num_grid_2, x_start, x_end, y_start, y_end);
113     this->current_grid = new RectangularSpatialGrid(num_grid_1, num_grid_2, x_start, x_end, y_start,
114     y_end);
114     this->potential_grid = new RectangularSpatialGrid(num_grid_1, num_grid_2, x_start, x_end, y_start,
115     y_end);
115 };

```

4.19.2.3 ~RectangularDomain()

RectangularDomain::~~RectangularDomain ()

Definition at line 116 of file rect_domain.cpp.

```

117 {
118     delete this->potential_grid;
119 };

```

4.19.3 Member Function Documentation

4.19.3.1 assign_initial_value()

```

void BaseDomain::assign_initial_value (
    int index_1,
    int index_2,
    std::complex< float > value ) [inherited]

```

Assign initial value.

Parameters

<i>index_1</i>	$x = x_start + index_1 * dx$
<i>index_2</i>	$y = y_start + index_2 * dy$
<i>value</i>	initial value at x, y

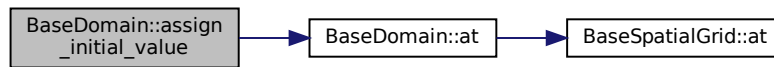
Definition at line 188 of file base_domain.cpp.

```

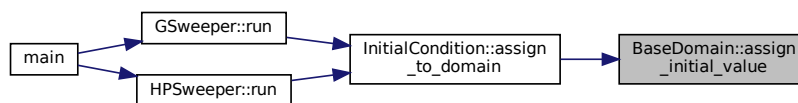
189 {
190     this->at(index_1, index_2, 0)->value = value;
191 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



4.19.3.2 assign_wave_function()

```

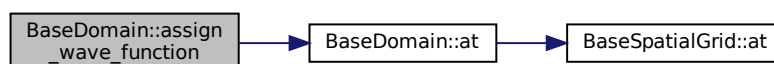
void BaseDomain::assign_wave_function (
    int index_1,
    int index_2,
    int time_index,
    std::complex< float > value ) [inherited]
  
```

Definition at line 192 of file `base_domain.cpp`.

```

193 {
194     this->at(index_1, index_2, time_index)->value = value;
195 }
  
```

Here is the call graph for this function:



4.19.3.3 at()

```
GridPoint * BaseDomain::at (
    int index_1,
    int index_2,
    int time_index ) [inherited]
```

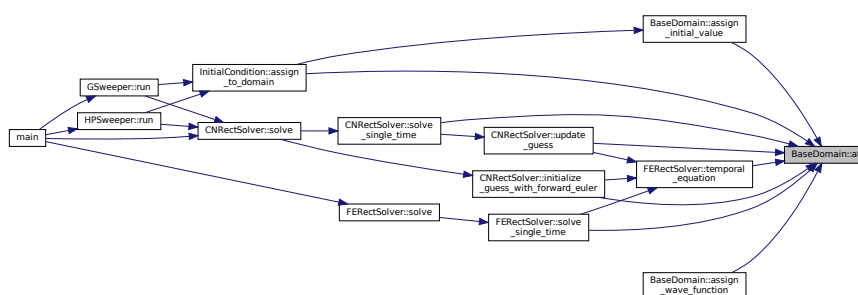
Definition at line 153 of file base_domain.cpp.

```
154 {
155     if (time_index == this->current_time_index)
156     {
157         return this->current_grid->at(index_1, index_2);
158     }
159     else if (time_index == this->current_time_index - 1)
160     {
161         return this->old_grid->at(index_1, index_2);
162     }
163     else
164     {
165         std::cerr << "error in base domin at function" << std::endl;
166         return this->current_grid->at(index_1, index_2);
167     }
168 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.19.3.4 generate_directory_name()

```
void BaseDomain::generate_directory_name (
    std::string info,
    bool print_info = true ) [inherited]
```

create directory and return directory name with "/" directory name : `"/results/%d-%m-%Y %H-%M-%S_"`+info for exmaple, `"/results/"`

Parameters

<i>info</i>	information about domain type and solver type
-------------	---

Returns

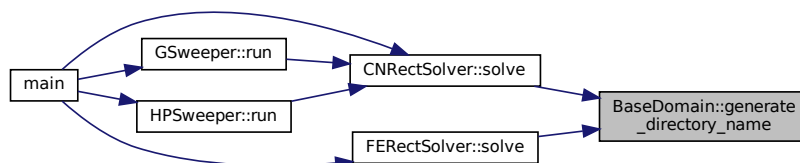
std::string directory name with "/"

Definition at line 203 of file base_domain.cpp.

```

204 {
205     auto t = std::time(nullptr);
206     auto tm = *std::localtime(&t);
207
208     std::ostringstream oss;
209     oss << std::put_time(&tm, "%Y-%m-%d-%H-%M-%S");
210     auto str = oss.str();
211     std::string directory_name = "../results/" + str + "_" + info;
212     bool created = fs::create_directory(directory_name.c_str());
213     if (print_info && created)
214     {
215         std::cout << "Created directory " << directory_name << std::endl;
216     }
217     // else if(print_info )
218     //TODO
219     else if (!created)
220     {
221         std::cout << "Creating directory failed" << std::endl;
222     }
223
224     this->PATH = directory_name + "/";
225 }
```

Here is the caller graph for this function:



4.19.3.5 generate_single_txt_file()

```

void RectangularDomain::generate_single_txt_file (
    std::string filename,
    bool cuda_mode = false )
```

Definition at line 152 of file rect_domain.cpp.

```

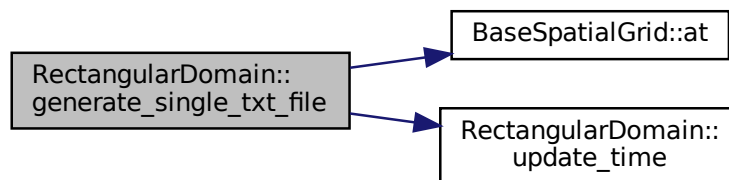
153 {
154     if (cuda_mode)
155     {
156         this->update_time(cuda_mode);
157     }
158     else
159     {
160         std::ofstream outfile(this->PATH + filename + ".txt");
161         outfile << "x, y, real, imag, magn, phase " << std::endl;
162         for (auto i = 0; i < num_grid_1; ++i)
163         {
```

```

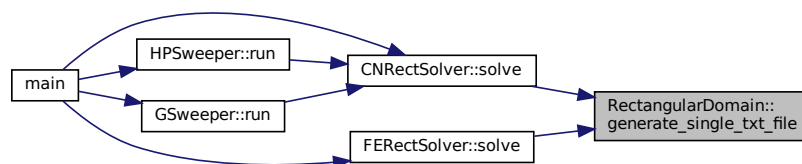
164         for (auto j = 0; j < num_grid_2; ++j)
165         {
166             float magnitude = std::abs(this->current_grid->at(i, j)->value);
167             float phase = std::arg(this->current_grid->at(i, j)->value);
168             outfile << this->current_grid->at(i, j)->x << ", " << this->current_grid->at(i, j)->y << ",
169 ";
169             outfile << this->current_grid->at(i, j)->value.real() << ", " << this->current_grid->at(i,
170 j)->value.imag() << ", ";
170             outfile << magnitude << ", " << phase;
171             outfile << std::endl;
172         }
173     }
174     outfile.close();
175     // After saving data, update domain
176     this->update_time();
177 }
178 };

```

Here is the call graph for this function:



Here is the caller graph for this function:



4.19.3.6 get_current_time_index()

```
int BaseDomain::get_current_time_index ( ) [inherited]
```

Definition at line 170 of file `base_domain.cpp`.

```

171 {
172     return this->current_time_index;
173 }

```

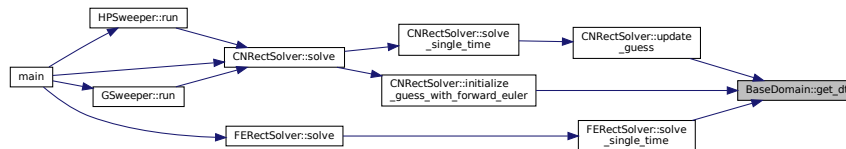
4.19.3.7 get_dt()

```
float BaseDomain::get_dt ( ) [inherited]
```

Definition at line 125 of file base_domain.cpp.

```
126 {
127     return this->dt;
128 }
```

Here is the caller graph for this function:



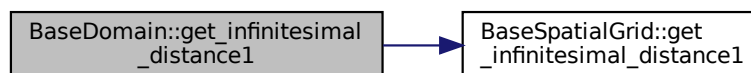
4.19.3.8 get_infinitesimal_distance1()

```
float BaseDomain::get_infinitesimal_distance1 ( ) [inherited]
```

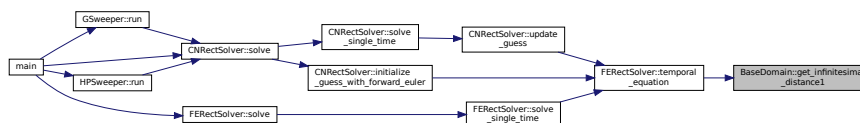
Definition at line 143 of file base_domain.cpp.

```
144 {
145     return this->old_grid->get_infinitesimal_distance1();
146 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



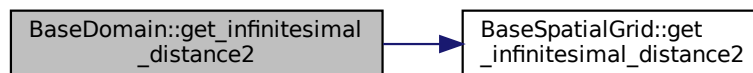
4.19.3.9 get_infinitesimal_distance2()

```
float BaseDomain::get_infinitesimal_distance2 ( ) [inherited]
```

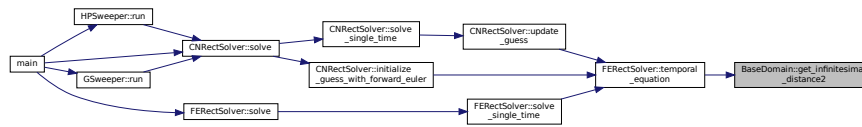
Definition at line 147 of file base_domain.cpp.

```
148 {
149     return this->old_grid->get_infinitesimal_distance2();
150 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



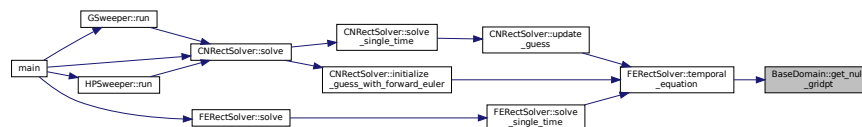
4.19.3.10 get_null_gridpt()

```
GridPoint * BaseDomain::get_null_gridpt ( ) [inherited]
```

Definition at line 176 of file base_domain.cpp.

```
177 {
178     return this->null_gridpt;
179 }
```

Here is the caller graph for this function:



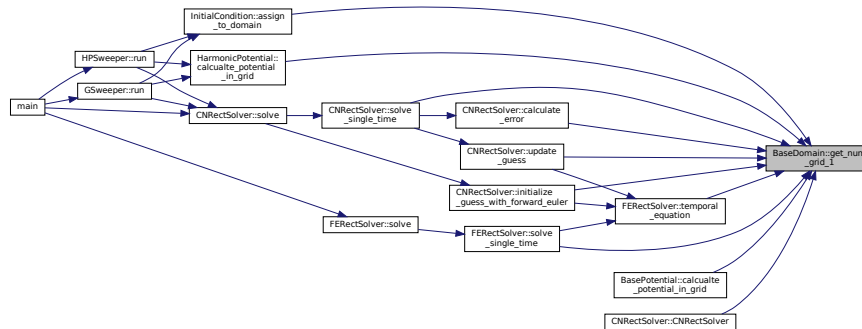
4.19.3.11 get_num_grid_1()

```
int BaseDomain::get_num_grid_1 ( ) [inherited]
```

Definition at line 134 of file base_domain.cpp.

```
135 {
136     return this->num_grid_1;
137 }
```

Here is the caller graph for this function:



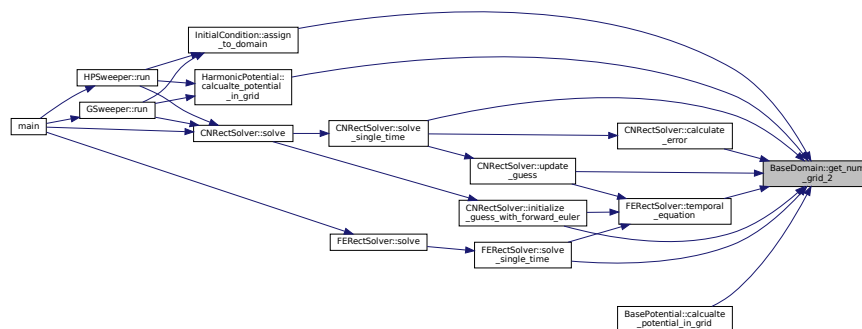
4.19.3.12 get_num_grid_2()

```
int BaseDomain::get_num_grid_2 ( ) [inherited]
```

Definition at line 138 of file base_domain.cpp.

```
139 {
140     return this->num_grid_2;
141 }
```

Here is the caller graph for this function:



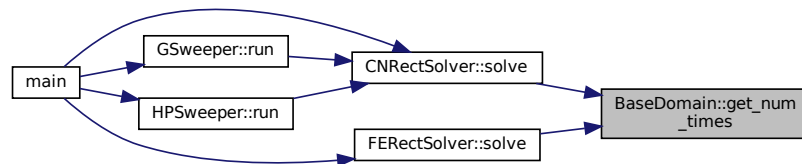
4.19.3.13 get_num_times()

```
int BaseDomain::get_num_times ( ) [inherited]
```

Definition at line 129 of file base_domain.cpp.

```
130 {
131     return this->num_times;
132 }
```

Here is the caller graph for this function:

**4.19.3.14 get_path()**

```
std::string BaseDomain::get_path ( ) [inherited]
```

Definition at line 106 of file base_domain.cpp.

```
107 {
108     return this->PATH;
109 }
```

4.19.3.15 get_t_end()

```
float BaseDomain::get_t_end ( ) [inherited]
```

Definition at line 121 of file base_domain.cpp.

```
122 {
123     return this->t_end;
124 }
```

4.19.3.16 get_t_start()

```
float BaseDomain::get_t_start ( ) [inherited]
```

Definition at line 116 of file base_domain.cpp.

```
117 {
118     return this->t_start;
119 }
```

4.19.3.17 get_x_end()

```
float RectangularDomain::get_x_end ( )
```

Definition at line 128 of file rect_domain.cpp.

```
129 {  
130     return this->x_end;  
131 }
```

4.19.3.18 get_x_start()

```
float RectangularDomain::get_x_start ( )
```

Definition at line 120 of file rect_domain.cpp.

```
121 {  
122     return this->x_start;  
123 }
```

4.19.3.19 get_y_end()

```
float RectangularDomain::get_y_end ( )
```

Definition at line 132 of file rect_domain.cpp.

```
133 {  
134     return this->y_end;  
135 }
```

4.19.3.20 get_y_start()

```
float RectangularDomain::get_y_start ( )
```

Definition at line 124 of file rect_domain.cpp.

```
125 {  
126     return this->y_start;  
127 }
```

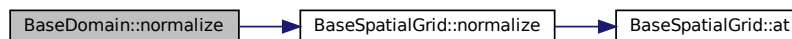
4.19.3.21 normalize()

```
void BaseDomain::normalize (
    int time_index ) [inherited]
```

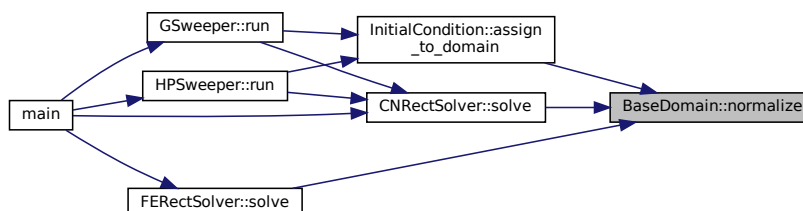
Definition at line 111 of file base_domain.cpp.

```
112 {
113     this->current_grid->normalize();
114 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



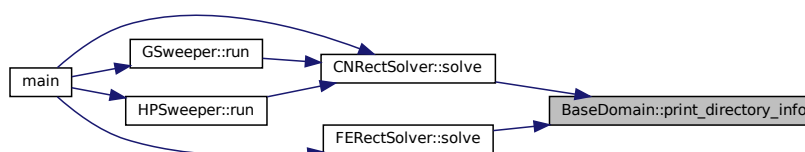
4.19.3.22 print_directory_info()

```
void BaseDomain::print_directory_info ( ) [inherited]
```

Definition at line 261 of file base_domain.cpp.

```
262 {
263     std::cout << this->num_times;
264     std::cout << " text files are generated in \n";
265     std::cout << this->PATH << std::endl;
266 }
```

Here is the caller graph for this function:



4.19.3.23 reset()

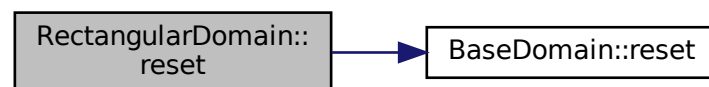
```
void RectangularDomain::reset ( ) [virtual]
```

Reimplemented from [BaseDomain](#).

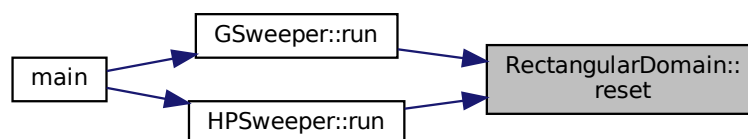
Definition at line 180 of file `rect_domain.cpp`.

```
181 {
182     BaseDomain::reset();
183     delete this->potential_grid;
184
185     this->old_grid = new RectangularSpatialGrid(num_grid_1, num_grid_2, x_start, x_end, y_start, y_end);
186     this->current_grid = new RectangularSpatialGrid(num_grid_1, num_grid_2, x_start, x_end, y_start,
187     y_end);
187     this->potential_grid = new RectangularSpatialGrid(num_grid_1, num_grid_2, x_start, x_end, y_start,
188     y_end);
188
189 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.19.3.24 time_at()

```
float BaseDomain::time_at (
    int time_index ) [inherited]
```

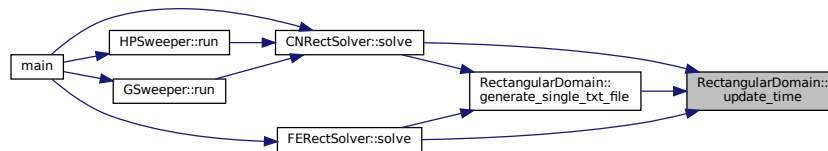
4.19.3.25 update_time()

```
void RectangularDomain::update_time (
    bool cuda_mode = false )
```

Definition at line 137 of file rect_domain.cpp.

```
138 {
139     if (cuda_mode)
140     {
141         this->current_time_index += 1;
142     }
143     else
144     {
145         this->current_time_index += 1;
146         delete (this->old_grid);
147         this->old_grid = this->current_grid;
148         this->current_grid = new RectangularSpatialGrid(num_grid_1, num_grid_2, x_start, x_end, y_start,
149             y_end);
149     }
150 }
```

Here is the caller graph for this function:



4.19.4 Member Data Documentation

4.19.4.1 current_grid

`BaseSpatialGrid*` `BaseDomain::current_grid` [protected], [inherited]

Definition at line 98 of file base_domain.h.

4.19.4.2 current_time_index

`int` `BaseDomain::current_time_index` = 0 [protected], [inherited]

Definition at line 102 of file base_domain.h.

4.19.4.3 dt

```
float BaseDomain::dt [protected], [inherited]
```

Definition at line 100 of file base_domain.h.

4.19.4.4 null_gridpt

```
GridPoint* BaseDomain::null_gridpt [protected], [inherited]
```

Definition at line 99 of file base_domain.h.

4.19.4.5 num_grid_1

```
int BaseDomain::num_grid_1 [protected], [inherited]
```

Definition at line 101 of file base_domain.h.

4.19.4.6 num_grid_2

```
int BaseDomain::num_grid_2 [protected], [inherited]
```

Definition at line 101 of file base_domain.h.

4.19.4.7 num_times

```
int BaseDomain::num_times [protected], [inherited]
```

Definition at line 101 of file base_domain.h.

4.19.4.8 old_grid

```
BaseSpatialGrid* BaseDomain::old_grid [protected], [inherited]
```

Definition at line 97 of file base_domain.h.

4.19.4.9 PATH

```
std::string BaseDomain::PATH [protected], [inherited]
```

Definition at line 103 of file base_domain.h.

4.19.4.10 potential_grid

```
RectangularSpatialGrid* RectangularDomain::potential_grid
```

Definition at line 48 of file rect_domain.h.

4.19.4.11 t_end

```
float BaseDomain::t_end [protected], [inherited]
```

Definition at line 100 of file base_domain.h.

4.19.4.12 t_start

```
float BaseDomain::t_start [protected], [inherited]
```

Definition at line 100 of file base_domain.h.

4.19.4.13 x_end

```
float RectangularDomain::x_end [private]
```

Definition at line 55 of file rect_domain.h.

4.19.4.14 x_start

```
float RectangularDomain::x_start [private]
```

Definition at line 54 of file rect_domain.h.

4.19.4.15 `y_end`

```
float RectangularDomain::y_end [private]
```

Definition at line 57 of file `rect_domain.h`.

4.19.4.16 `y_start`

```
float RectangularDomain::y_start [private]
```

Definition at line 56 of file `rect_domain.h`.

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

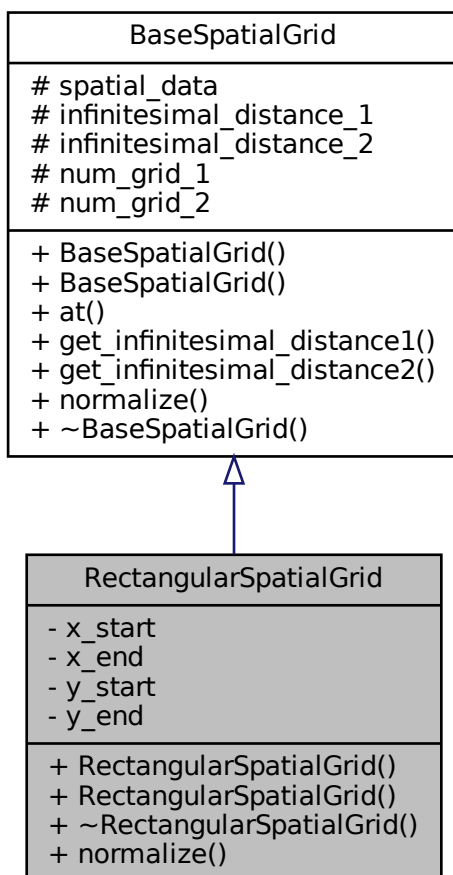
- [/home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/domain/`rect_domain.h`](#)
- [/home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/domain/`rect_domain.cpp`](#)

4.20 RectangularSpatialGrid Class Reference

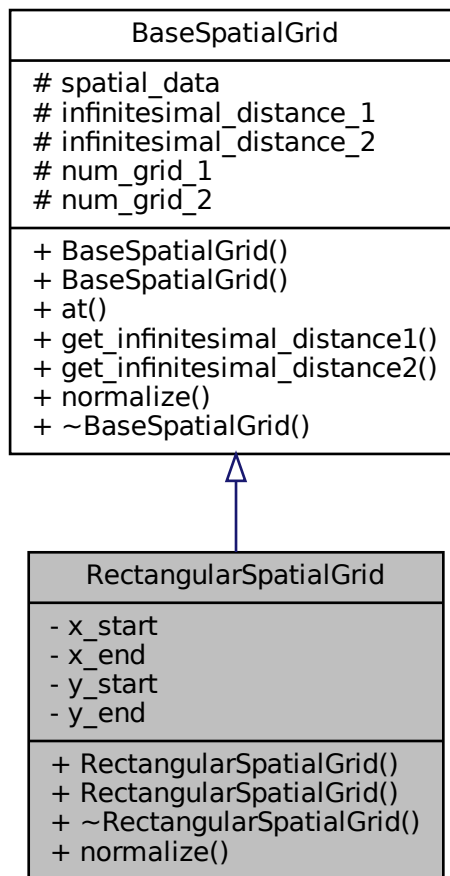
Rectangular Spatial Grid class for single time step.

```
#include <rect_domain.h>
```

Inheritance diagram for RectangularSpatialGrid:



Collaboration diagram for RectangularSpatialGrid:



Public Member Functions

- [RectangularSpatialGrid \(\)](#)=default
- [RectangularSpatialGrid \(int num_grid_1, int num_grid_2, float x_start, float x_end, float y_start, float y_end\)](#)
Construct a new Rectangular Spatial Grid:: Rectangular Spatial Grid object.
- [~RectangularSpatialGrid \(\)](#)
- void [normalize \(\)](#)
- [GridPoint * at \(int index_1, int index_2\)](#)
- float [get_infinitesimal_distance1 \(\)](#)
- float [get_infinitesimal_distance2 \(\)](#)

Protected Attributes

- `std::vector< std::vector< GridPoint > > spatial_data`
- float [infinitesimal_distance_1](#)
- float [infinitesimal_distance_2](#)
- int [num_grid_1](#)
- int [num_grid_2](#)

Private Attributes

- float [x_start](#)
- float [x_end](#)
- float [y_start](#)
- float [y_end](#)

4.20.1 Detailed Description

Rectangular Spatial Grid class for single time step.

Definition at line 18 of file `rect_domain.h`.

4.20.2 Constructor & Destructor Documentation

4.20.2.1 RectangularSpatialGrid() [1/2]

```
RectangularSpatialGrid::RectangularSpatialGrid ( ) [default]
```

4.20.2.2 RectangularSpatialGrid() [2/2]

```
RectangularSpatialGrid::RectangularSpatialGrid (
    int num_grid_1,
    int num_grid_2,
    float x_start,
    float x_end,
    float y_start,
    float y_end )
```

Construct a new Rectangular Spatial Grid:: Rectangular Spatial Grid object.

Parameters

<i>num_grid_1</i>	Number of grids in x axis
<i>num_grid_2</i>	Number of grids in y axis
<i>x_start</i>	Start point of x axis
<i>x_end</i>	End point of x axis
<i>y_start</i>	Start point of y axis
<i>y_end</i>	Start point of x axis

Definition at line 26 of file `rect_domain.cpp`.

```
32 : BaseSpatialGrid(num_grid_1, num_grid_2)
```

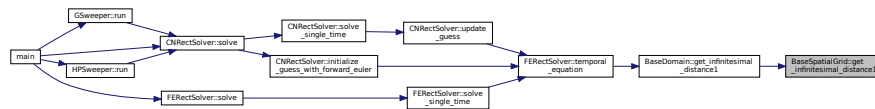

4.20.3.2 get_infinitesimal_distance1()

```
float BaseSpatialGrid::get_infinitesimal_distance1 ( ) [inherited]
```

Definition at line 64 of file base_domain.cpp.

```
65 {
66     return this->infinitesimal_distance_1;
67 }
```

Here is the caller graph for this function:



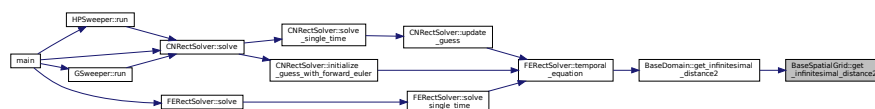
4.20.3.3 get_infinitesimal_distance2()

```
float BaseSpatialGrid::get_infinitesimal_distance2 ( ) [inherited]
```

Definition at line 70 of file base_domain.cpp.

```
71 {
72     return this->infinitesimal_distance_2;
73 }
```

Here is the caller graph for this function:



4.20.3.4 normalize()

```
void RectangularSpatialGrid::normalize ( )
```

Definition at line 53 of file rect_domain.cpp.

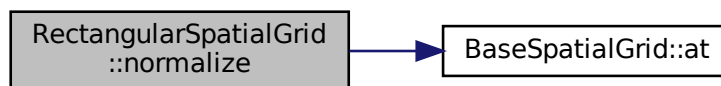
```
54 {
55     float sum = 0.;
56     for (auto i = 0; i < this->num_grid_1; ++i)
57     {
58         for (auto j = 0; j < this->num_grid_2; ++j)
59         {
60             auto wave_func = this->at(i, j)->value;
61             sum += float(std::pow(std::abs(wave_func), 2));
62         }
63     }
```

```

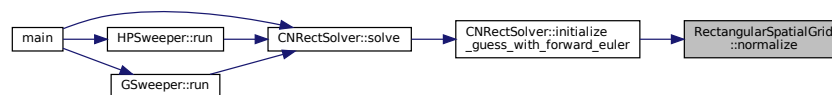
64     sum = std::sqrt(sum * this->infinitesimal_distance_1 * this->infinitesimal_distance_2);
65     for (auto i = 0; i < this->num_grid_1; ++i)
66     {
67         for (auto j = 0; j < this->num_grid_2; ++j)
68         {
69             this->at(i, j)->value /= sum;
70         }
71     }
72 }
73 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



4.20.4 Member Data Documentation

4.20.4.1 infinitesimal_distance_1

```
float BaseSpatialGrid::infinitesimal_distance_1 [protected], [inherited]
```

Definition at line 59 of file `base_domain.h`.

4.20.4.2 infinitesimal_distance_2

```
float BaseSpatialGrid::infinitesimal_distance_2 [protected], [inherited]
```

Definition at line 59 of file `base_domain.h`.

4.20.4.3 num_grid_1

```
int BaseSpatialGrid::num_grid_1 [protected], [inherited]
```

Definition at line 60 of file base_domain.h.

4.20.4.4 num_grid_2

```
int BaseSpatialGrid::num_grid_2 [protected], [inherited]
```

Definition at line 60 of file base_domain.h.

4.20.4.5 spatial_data

```
std::vector<std::vector<GridPoint> > BaseSpatialGrid::spatial_data [protected], [inherited]
```

Definition at line 58 of file base_domain.h.

4.20.4.6 x_end

```
float RectangularSpatialGrid::x_end [private]
```

Definition at line 28 of file rect_domain.h.

4.20.4.7 x_start

```
float RectangularSpatialGrid::x_start [private]
```

Definition at line 27 of file rect_domain.h.

4.20.4.8 y_end

```
float RectangularSpatialGrid::y_end [private]
```

Definition at line 30 of file rect_domain.h.

4.20.4.9 y_start

```
float RectangularSpatialGrid::y_start [private]
```

Definition at line 29 of file rect_domain.h.

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

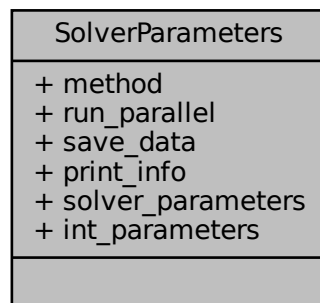
- [/home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/domain/rect_domain.h](#)
- [/home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/domain/rect_domain.cpp](#)

4.21 SolverParameters Class Reference

Solver branch parameters containing information about solver.

```
#include <parameters.h>
```

Collaboration diagram for SolverParameters:



Public Attributes

- `std::string method = std::string("")`
- `bool run_parallel`
- `bool save_data`
- `bool print_info`
- `std::map< std::string, float > solver_parameters`
- `std::map< std::string, int > int_parameters`

4.21.1 Detailed Description

Solver branch parameters containing information about solver.

Definition at line 69 of file parameters.h.

4.21.2 Member Data Documentation

4.21.2.1 int_parameters

```
std::map<std::string, int> SolverParameters::int_parameters
```

Definition at line 77 of file parameters.h.

4.21.2.2 method

```
std::string SolverParameters::method = std::string("")
```

Definition at line 72 of file parameters.h.

4.21.2.3 print_info

```
bool SolverParameters::print_info
```

Definition at line 75 of file parameters.h.

4.21.2.4 run_parallel

```
bool SolverParameters::run_parallel
```

Definition at line 73 of file parameters.h.

4.21.2.5 save_data

```
bool SolverParameters::save_data
```

Definition at line 74 of file parameters.h.

4.21.2.6 solver_parameters

```
std::map<std::string, float> SolverParameters::solver_parameters
```

Definition at line 76 of file parameters.h.

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

- /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/configuration/[parameters.h](#)

Chapter 5

File Documentation

5.1 /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/CMakeLists.txt File Reference

Functions

- [cmake_minimum_required](#) (VERSION 3.5) [enable_language\(CUDA\)](#) [file\(GLOB_RECURSE SRC_FILES LIST_DIRECTORIES false *.h *.cpp *.cu *cuh\)](#) [find_package\(MPI REQUIRED\)](#) [include_directories\(\\$](#)
- [target_link_libraries](#) (\${BINARY}_run \${MPI_LIBRARIES}) [target_link_libraries\(\\$](#)

5.1.1 Function Documentation

5.1.1.1 cmake_minimum_required()

```
cmake_minimum_required (
    VERSION 3. 5 )
```

Definition at line 1 of file CMakeLists.txt.

```
4         {CMAKE_PROJECT_NAME} )
5 SET( EXECUTABLE_OUTPUT_PATH ${CMAKE_SOURCE_DIR}/build )
6 SET( LIBRARY_OUTPUT_PATH ${CMAKE_SOURCE_DIR}/build )
7
8 file(GLOB_RECURSE SRC_FILES LIST_DIRECTORIES false *.h *.cpp *.cu *cuh)
9 ADD_EXECUTABLE( ${BINARY}_run ${SRC_FILES} )
10 ADD_LIBRARY( ${BINARY}_lib STATIC ${SRC_FILES} )
11
12 #MPI linking
13 find_package(MPI REQUIRED)
14 include_directories(${MPI_INCLUDE_PATH})
```

5.1.1.2 target_link_libraries()

```
target_link_libraries (
    ${BINARY}_run ${MPI_LIBRARIES} )
```

Definition at line 15 of file CMakeLists.txt.

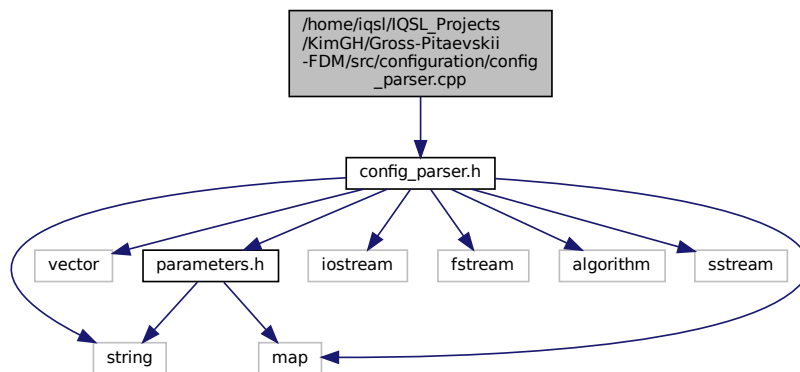
```
15         {BINARY}_run ${MPI_LIBRARIES})
16 target_link_libraries(${BINARY}_lib ${MPI_LIBRARIES})
```

5.2 /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FD↵ M/src/configuration/config_parser.cpp File Reference

Configuration Parser class implementation.

```
#include "config_parser.h"
```

Include dependency graph for config_parser.cpp:



5.2.1 Detailed Description

Configuration Parser class implementation.

Author

Gyeonghun Kim, Minyoung Kim

Version

0.1

Date

2022-06-09

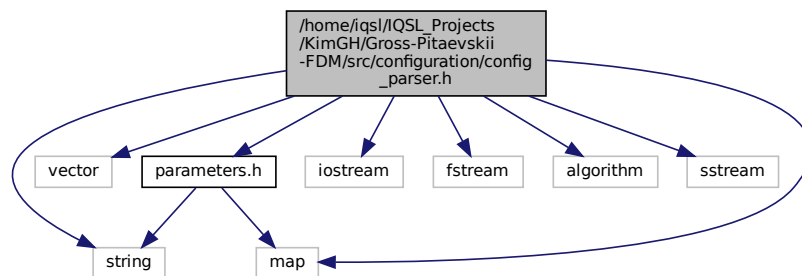
Copyright

Copyright (c) 2022

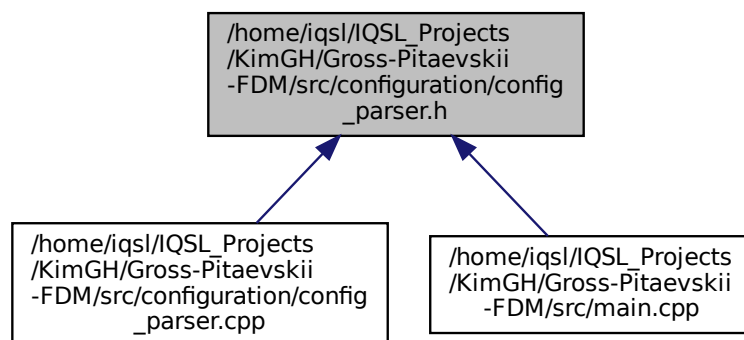
5.3 /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/configuration/config_parser.h File Reference

Configuration Parser Class header.

```
#include <string>
#include <vector>
#include <map>
#include <iostream>
#include <fstream>
#include <algorithm>
#include <sstream>
#include "parameters.h"
Include dependency graph for config_parser.h:
```



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



Classes

- class [ConfigParser](#)
Configuration parser.

5.3.1 Detailed Description

Configuration Parser Class header.

Author

Gyeonghun Kim, Minyoung Kim

Version

0.1

Date

2022-06-09

Copyright

Copyright (c) 2022

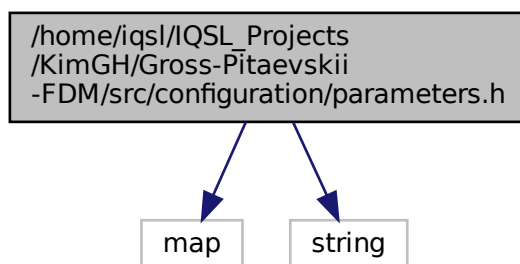
5.4 /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FD↵ M/src/configuration/parameters.h File Reference

Header for the Parameter classes.

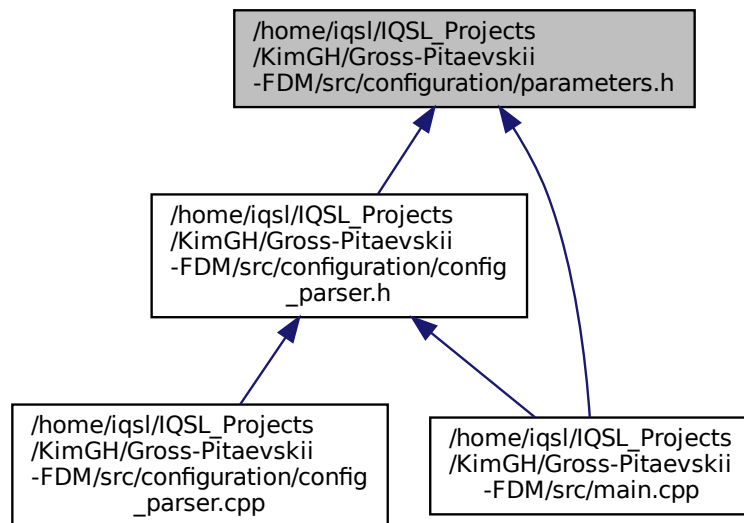
```
#include <map>
```

```
#include <string>
```

Include dependency graph for parameters.h:



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



Classes

- class [MainParameters](#)
Main branch parameters containing information about overall calculation.
- class [DomainParameters](#)
Domain branch parameters containing information about domain.
- class [InitialConditionParameters](#)
Initial condition branch parameters containing information about initial condition.
- class [EquationParameters](#)
Equation branch parameters containing information about equation.
- class [SolverParameters](#)
Solver branch parameters containing information about solver.
- class [Parameters](#)
[Parameters](#) containing configuration name and all other branched parameters.

5.4.1 Detailed Description

Header for the Parameter classes.

Author

Gyeonghun Kim, Minyoung Kim

Version

0.1

Date

2022-06-09

Copyright

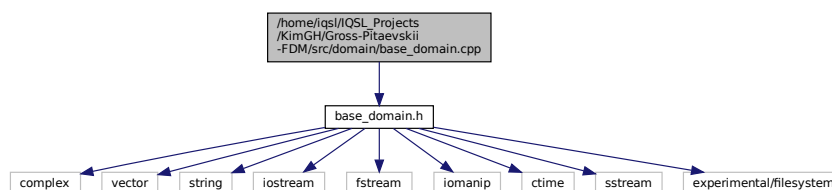
Copyright (c) 2022

5.5 /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FD↩ M/src/domain/base_domain.cpp File Reference

Implementation of methods in the Base Domain class.

```
#include "base_domain.h"
```

Include dependency graph for base_domain.cpp:



5.5.1 Detailed Description

Implementation of methods in the Base Domain class.

Author

Minyoung Kim, Gyeonghun Kim

Version

0.1

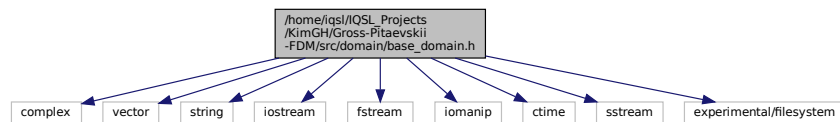
Date

2022-06-03

Copyright

Copyright (c) 2022

```
#include <complex>
#include <vector>
#include <string>
#include <iostream>
#include <fstream>
#include <iomanip>
#include <ctime>
#include <sstream>
#include <experimental/filesystem>
Include dependency graph for base_domain.h:
```



```

graph TD
    A["/home/lqsl/IQSL_Projects/KimGH/Gross-Piteavskii/FDM/src/domain_base_domain.h"]
    B["/home/lqsl/IQSL_Projects/KimGH/Gross-Piteavskii/FDM/src/domain_base_domain.cpp"]
    C["/home/lqsl/IQSL_Projects/KimGH/Gross-Piteavskii/FDM/src/domain_rect_domain.h"]
    D["/home/lqsl/IQSL_Projects/KimGH/Gross-Piteavskii/FDM/src/domain_rect_domain.cpp"]
    E["/home/lqsl/IQSL_Projects/KimGH/Gross-Piteavskii/FDM/src/initial_condition/initial_condition.h"]
    F["/home/lqsl/IQSL_Projects/KimGH/Gross-Piteavskii/FDM/src/initial_condition/initial_condition.cpp"]
    G["/home/lqsl/IQSL_Projects/KimGH/Gross-Piteavskii/FDM/src/solver/base_solver.h"]
    H["/home/lqsl/IQSL_Projects/KimGH/Gross-Piteavskii/FDM/src/solver/base_solver.cpp"]
    I["/home/lqsl/IQSL_Projects/KimGH/Gross-Piteavskii/FDM/src/solver/serial_solver/forward_eulerfe_rect_solver.h"]
    J["/home/lqsl/IQSL_Projects/KimGH/Gross-Piteavskii/FDM/src/solver/serial_solver/forward_eulerfe_rect_solver.cpp"]
    K["/home/lqsl/IQSL_Projects/KimGH/Gross-Piteavskii/FDM/src/solver/serial_solver/forward_eulerfe_rect_solver.h"]
    L["/home/lqsl/IQSL_Projects/KimGH/Gross-Piteavskii/FDM/src/solver/serial_solver/forward_eulerfe_rect_solver.cpp"]
    M["/home/lqsl/IQSL_Projects/KimGH/Gross-Piteavskii/FDM/src/solver/serial_solver/forward_eulerfe_rect_solver.h"]
    N["/home/lqsl/IQSL_Projects/KimGH/Gross-Piteavskii/FDM/src/solver/serial_solver/forward_eulerfe_rect_solver.cpp"]
    O["/home/lqsl/IQSL_Projects/KimGH/Gross-Piteavskii/FDM/src/solver/serial_solver/forward_eulerfe_rect_solver.h"]
    P["/home/lqsl/IQSL_Projects/KimGH/Gross-Piteavskii/FDM/src/solver/serial_solver/forward_eulerfe_rect_solver.cpp"]
    Q["/home/lqsl/IQSL_Projects/KimGH/Gross-Piteavskii/FDM/src/solver/serial_solver/forward_eulerfe_rect_solver.h"]
    R["/home/lqsl/IQSL_Projects/KimGH/Gross-Piteavskii/FDM/src/solver/serial_solver/forward_eulerfe_rect_solver.cpp"]
    S["/home/lqsl/IQSL_Projects/KimGH/Gross-Piteavskii/FDM/src/solver/serial_solver/forward_eulerfe_rect_solver.h"]
    T["/home/lqsl/IQSL_Projects/KimGH/Gross-Piteavskii/FDM/src/solver/serial_solver/forward_eulerfe_rect_solver.cpp"]
    U["/home/lqsl/IQSL_Projects/KimGH/Gross-Piteavskii/FDM/src/solver/serial_solver/forward_eulerfe_rect_solver.h"]
    V["/home/lqsl/IQSL_Projects/KimGH/Gross-Piteavskii/FDM/src/solver/serial_solver/forward_eulerfe_rect_solver.cpp"]
    W["/home/lqsl/IQSL_Projects/KimGH/Gross-Piteavskii/FDM/src/solver/serial_solver/forward_eulerfe_rect_solver.h"]
    X["/home/lqsl/IQSL_Projects/KimGH/Gross-Piteavskii/FDM/src/solver/serial_solver/forward_eulerfe_rect_solver.cpp"]
    Y["/home/lqsl/IQSL_Projects/KimGH/Gross-Piteavskii/FDM/src/solver/serial_solver/forward_eulerfe_rect_solver.h"]
    Z["/home/lqsl/IQSL_Projects/KimGH/Gross-Piteavskii/FDM/src/solver/serial_solver/forward_eulerfe_rect_solver.cpp"]

    A --> B
    A --> C
    A --> E
    A --> G
    A --> I
    A --> K
    A --> M
    A --> O
    A --> Q
    A --> S
    A --> U
    A --> W
    A --> Y
    A --> Z
    B --> C
    B --> E
    B --> G
    B --> I
    B --> K
    B --> M
    B --> O
    B --> Q
    B --> S
    B --> U
    B --> W
    B --> Y
    B --> Z
    C --> D
    C --> E
    C --> G
    C --> I
    C --> K
    C --> M
    C --> O
    C --> Q
    C --> S
    C --> U
    C --> W
    C --> Y
    C --> Z
    D --> E
    D --> G
    D --> I
    D --> K
    D --> M
    D --> O
    D --> Q
    D --> S
    D --> U
    D --> W
    D --> Y
    D --> Z
    E --> F
    E --> G
    E --> I
    E --> K
    E --> M
    E --> O
    E --> Q
    E --> S
    E --> U
    E --> W
    E --> Y
    E --> Z
    F --> G
    F --> I
    F --> K
    F --> M
    F --> O
    F --> Q
    F --> S
    F --> U
    F --> W
    F --> Y
    F --> Z
    G --> H
    G --> I
    G --> K
    G --> M
    G --> O
    G --> Q
    G --> S
    G --> U
    G --> W
    G --> Y
    G --> Z
    H --> I
    H --> K
    H --> M
    H --> O
    H --> Q
    H --> S
    H --> U
    H --> W
    H --> Y
    H --> Z
    I --> J
    I --> K
    I --> M
    I --> O
    I --> Q
    I --> S
    I --> U
    I --> W
    I --> Y
    I --> Z
    J --> K
    J --> M
    J --> O
    J --> Q
    J --> S
    J --> U
    J --> W
    J --> Y
    J --> Z
    K --> L
    K --> M
    K --> O
    K --> Q
    K --> S
    K --> U
    K --> W
    K --> Y
    K --> Z
    L --> M
    L --> O
    L --> Q
    L --> S
    L --> U
    L --> W
    L --> Y
    L --> Z
    M --> N
    M --> O
    M --> Q
    M --> S
    M --> U
    M --> W
    M --> Y
    M --> Z
    N --> O
    N --> Q
    N --> S
    N --> U
    N --> W
    N --> Y
    N --> Z
    O --> P
    O --> Q
    O --> S
    O --> U
    O --> W
    O --> Y
    O --> Z
    P --> Q
    P --> S
    P --> U
    P --> W
    P --> Y
    P --> Z
    Q --> R
    Q --> S
    Q --> U
    Q --> W
    Q --> Y
    Q --> Z
    R --> S
    R --> U
    R --> W
    R --> Y
    R --> Z
    S --> T
    S --> U
    S --> W
    S --> Y
    S --> Z
    T --> U
    T --> W
    T --> Y
    T --> Z
    U --> V
    U --> W
    U --> Y
    U --> Z
    V --> W
    V --> Y
    V --> Z
    W --> X
    W --> Y
    W --> Z
    X --> Y
    X --> Z
    Y --> Z
  
```

- class `GridPoint`
Grid point class (single point)
- class `BaseSpatialGrid`
Base Spatial Grid class for single time step.
- class `BaseDomain`
Base domain class containing multiple time steps and export methods.

5.6.1 Detailed Description

Header for Base domain class.

Author

Minyoung Kim, Gyeonghun Kim

Version

0.1

Date

2022-06-03

Copyright

Copyright (c) 2022

5.7 `/home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FD` `M/src/domain/rect_domain.cpp` File Reference

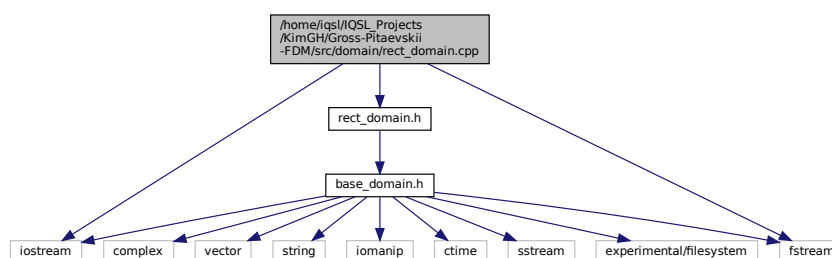
Implementation of methods in the Rectangular Domain class.

```
#include "rect_domain.h"
```

```
#include <fstream>
```

```
#include <iostream>
```

Include dependency graph for `rect_domain.cpp`:



Classes

- class [RectangularSpatialGrid](#)
Rectangular Spatial Grid class for single time step.
- class [RectangularDomain](#)
Rectangular domain containing multiple timesteps.

5.8.1 Detailed Description

Header for Rectangular domain class.

Author

Minyoung Kim, Gyeonghun Kim

Version

0.1

Date

2022-06-02

Copyright

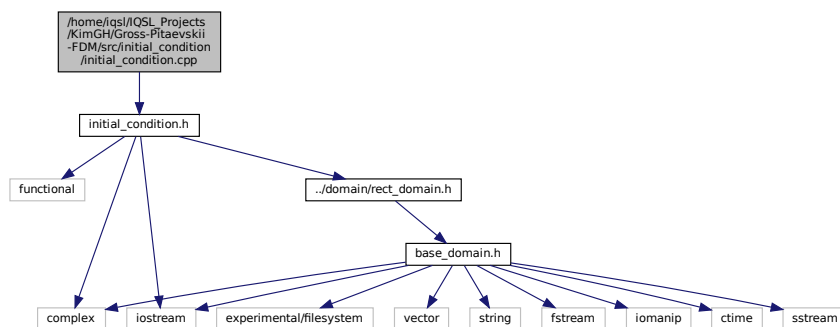
Copyright (c) 2022

5.9 `/home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/initial_condition/initial_condition.cpp` File Reference

Implementation of initial condition class.

```
#include "initial_condition.h"
```

Include dependency graph for `initial_condition.cpp`:



5.9.1 Detailed Description

Implementation of initial condition class.

Author

Gyeonghun Kim, Minyoung Kim

Version

0.1

Date

2022-06-04

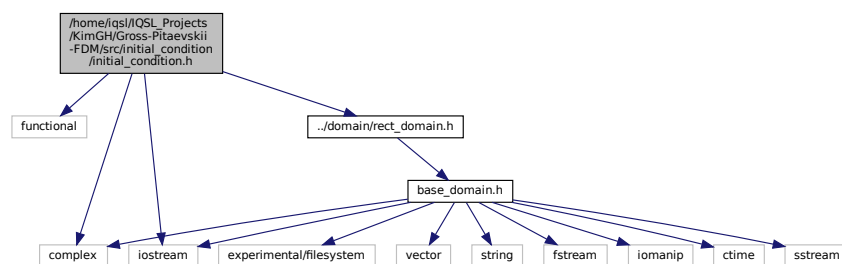
Copyright

Copyright (c) 2022

5.10 /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/initial_condition/initial_condition.h File Reference

Header for the initial condition class.

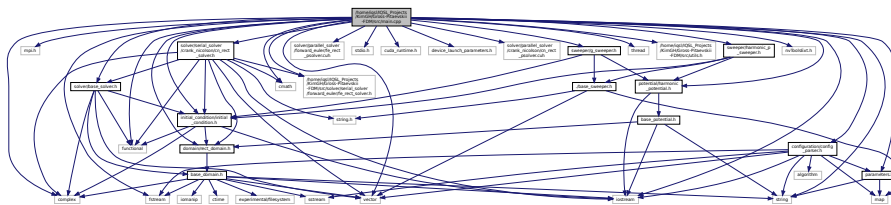
```
#include <functional>
#include <complex>
#include <iostream>
#include "../domain/rect_domain.h"
Include dependency graph for initial_condition.h:
```



5.11 /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/main.cpp File Reference

```
#include <iostream>
#include <mpi.h>
#include "domain/rect_domain.h"
#include "initial_condition/initial_condition.h"
#include "potential/harmonic_potential.h"
#include "solver/base_solver.h"
#include "solver/parallel_solver/forward_euler/fe_rect_psolver.cuh"
#include "solver/parallel_solver/crank_nicolson/cn_rect_psolver.cuh"
#include "solver/serial_solver/crank_nicolson/cn_rect_solver.h"
#include "configuration/config_parser.h"
#include "configuration/parameters.h"
#include "sweeper/g_sweeper.h"
#include "sweeper/harmonic_p_sweeper.h"
```

Include dependency graph for main.cpp:



Functions

- int [main](#) (int argc, char *argv[])

5.11.1 Function Documentation

5.11.1.1 main()

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

Definition at line 18 of file main.cpp.

```
19 {
20     //Initialize MPI
21     MPI_Init(&argc, &argv);
22     int rank, size;
23     MPI_Comm comm = MPI_COMM_WORLD;
24     MPI_Comm_rank(comm, &rank);
25     MPI_Comm_size(comm, &size);
26     Parameters parameters;
27
28     // If argc == 1: no input configuration => use default
29     if (argc == 1)
30     {
```

```

31     parameters = ConfigParser::get_default();
32 }
33 // argc != 1 and argc != 2 => exception
34 else if(argc != 2){
35     std::cerr << "Input config file required. " << std::endl;
36 }
37 // else: use given configuration
38 else{
39     std::string config_filename = std::string(argv[1]);
40     std::size_t dir_pos = config_filename.find_last_of("/");
41     std::string dir = config_filename.substr(0, dir_pos);
42     std::string config_name_extension = config_filename.substr(dir_pos + 1,
config_filename.length());
43     std::string config_name = config_name_extension.substr(0, config_name_extension.length() - 4);
44
45     parameters = ConfigParser::parse(config_name, config_filename);
46 }
47
48 // Non rectangular domain is not ready in this project, but leave this part for further development
49 if (parameters.domain_parameters.domain_type == "rectangular")
50 {
51     // Single calculation
52     if (parameters.main_parameters.calculation_type == "single")
53     {
54         // Prepare domain
55         RectangularDomain *domain = new RectangularDomain(parameters.domain_parameters.n_x,
parameters.domain_parameters.n_y,
parameters.domain_parameters.time_start,
parameters.domain_parameters.time_end,
parameters.domain_parameters.n_time,
parameters.domain_parameters.spatial_parameters["x_start"],
parameters.domain_parameters.spatial_parameters["x_end"],
parameters.domain_parameters.spatial_parameters["y_start"],
parameters.domain_parameters.spatial_parameters["y_end"]);
56
57         // Prepare initialcondition
58         std::function<std::complex<float>>(float, float)> initial_cond_function;
59         if (parameters.init_cond_parameters.init_cond_type == "singlegaussian")
60         {
61             auto sigma_x = parameters.init_cond_parameters.init_cond_parameters["sigma_x"];
62             auto sigma_y = parameters.init_cond_parameters.init_cond_parameters["sigma_y"];
63             auto center_x = parameters.init_cond_parameters.init_cond_parameters["center_x"];
64             auto center_y = parameters.init_cond_parameters.init_cond_parameters["center_y"];
65             initial_cond_function = [center_x, center_y, sigma_x, sigma_y](float x, float y)
66             {
67                 return std::complex<float>{
68                     float(1.) * expf(-(x - center_x) * (x - center_x) / (sigma_x * sigma_x) + (y -
center_y) * (y - center_y) / (sigma_y * sigma_y))
69                 };
70             };
71             auto *initial_condition = new InitialCondition(initial_cond_function);
72             initial_condition->assign_to_domain(domain);
73         }
74         // Library user could add case in here if they want to use non-gaussian initial condition.
75         else
76         {
77             std::cerr << "Unexpected initial condition " << std::endl;
78         }
79
80         // Prepare potential
81         if (parameters.equation_parameters.potential_type == "harmonic")
82         {
83             auto omega_x = parameters.equation_parameters.potential_parameters["omega_x"];
84             auto omega_y = parameters.equation_parameters.potential_parameters["omega_y"];
85             auto *potential = new HarmonicPotential(omega_x, omega_y);
86             potential->calcualte_potential_in_grid(domain);
87         }
88         // Library user could add case in here if they want to use non-harmonic potential
89         else
90         {
91             std::cerr << "Unexpected initial condition " << std::endl;
92         }
93
94         float g = parameters.equation_parameters.g;
95
96         bool save_data = parameters.solver_parameters.save_data;
97         bool print_info = parameters.solver_parameters.print_info;
98
99         // Solve equation in given method from configuration
100         if (parameters.solver_parameters.method == "cranknicolson")
101         {
102             if(parameters.solver_parameters.run_parallel){
103                 float converge_crit =
parameters.solver_parameters.solver_parameters["converge_crit"];

```

```

111         int max_iter = parameters.solver_parameters.int_parameters["max_iter"];
112         int cuda_device = parameters.solver_parameters.int_parameters["cuda_device"];
113         CNRectPSolver solver = CNRectPSolver(g, domain, cuda_device);
114         solver.solve(converge_crit, max_iter, std::to_string(rank), print_info, save_data);
115     }
116     else{
117         float converge_crit =
parameters.solver_parameters.solver_parameters["converge_crit"];
118         int max_iter = parameters.solver_parameters.int_parameters["max_iter"];
119         CNRectPSolver solver = CNRectPSolver(g, domain);
120         //std::cout<<"save_data " <<"save_data"<<std::endl;
121         solver.solve(converge_crit, max_iter, std::to_string(rank), print_info, save_data);
122     }
123 }
124 else if (parameters.solver_parameters.method == "forwardeuler"){
125     if (parameters.solver_parameters.run_parallel)
126     {
127         int cuda_device = parameters.solver_parameters.int_parameters["cuda_device"];
128         FERectPSolver solver = FERectPSolver(g, domain, cuda_device);
129         solver.solve(std::to_string(rank), print_info, save_data);
130     }
131     else
132     {
133         FERectSolver solver = FERectSolver(g, domain);
134         solver.solve(std::to_string(rank), print_info, save_data);
135     }
136 }
137 }
138 else if (parameters.main_parameters.calculation_type == "g_sweep")
139 {
140     // Prepare G sweep
141     GSweeper gSweeper = GSweeper(parameters.main_parameters.float_parameters["sweep_start"],
142                                     parameters.main_parameters.float_parameters["sweep_end"],
143                                     parameters.main_parameters.int_parameters["sweep_count"],
144                                     bool(parameters.main_parameters.int_parameters["end_point"]));
145     // Initialize domain
146     RectangularDomain *domain = new RectangularDomain(parameters.domain_parameters.n_x,
147                                                         parameters.domain_parameters.n_y,
148                                                         parameters.domain_parameters.time_start,
149                                                         parameters.domain_parameters.time_end,
150                                                         parameters.domain_parameters.n_time,
151                                                         parameters.domain_parameters.spatial_parameters["x_start"],
152                                                         parameters.domain_parameters.spatial_parameters["x_end"],
153                                                         parameters.domain_parameters.spatial_parameters["y_start"],
154                                                         parameters.domain_parameters.spatial_parameters["y_end"]);
155
156     // Prepare initial condition
157     std::function<std::complex<float>(float, float)> initial_cond_function;
158     if (parameters.init_cond_parameters.init_cond_type == "singlegaussian")
159     {
160         // prepare initial condition
161         auto sigma_x = parameters.init_cond_parameters.init_cond_parameters["sigma_x"];
162         auto sigma_y = parameters.init_cond_parameters.init_cond_parameters["sigma_y"];
163         auto center_x = parameters.init_cond_parameters.init_cond_parameters["center_x"];
164         auto center_y = parameters.init_cond_parameters.init_cond_parameters["center_y"];
165         initial_cond_function = [center_x, center_y, sigma_x, sigma_y](float x, float y)
166         {
167             return std::complex<float>{
168                 float(1.) * expf(-((x - center_x) * (x - center_x) / (sigma_x * sigma_x) + (y -
169 center_y) * (y - center_y) / (sigma_y * sigma_y))));
170             };
171         auto *initial_condition = new InitialCondition(initial_cond_function);
172         initial_condition->assign_to_domain(domain);
173     }
174     // Prepare potential
175     if (parameters.equation_parameters.potential_type == "harmonic")
176     {
177         auto omega_x = parameters.equation_parameters.potential_parameters["omega_x"];
178         auto omega_y = parameters.equation_parameters.potential_parameters["omega_y"];
179         auto *potential = new HarmonicPotential(omega_x, omega_y);
180         potential->calcuale_potential_in_grid(domain);
181
182         bool save_data = parameters.solver_parameters.save_data;
183         bool print_info = parameters.solver_parameters.print_info;
184         bool cuda_use = (bool)parameters.main_parameters.int_parameters["cuda_use"];
185         bool mpi_use = (bool)parameters.main_parameters.int_parameters["mpi_use"];
186         if (cuda_use)
187         {
188             // setup cuda if cuda_use
189             gSweeper.set_CUDA_info(parameters.main_parameters.int_parameters["gpu_count"],
190                                     parameters.main_parameters.int_parameters["calculation_per_gpu"]);
191         }
192     }

```

```

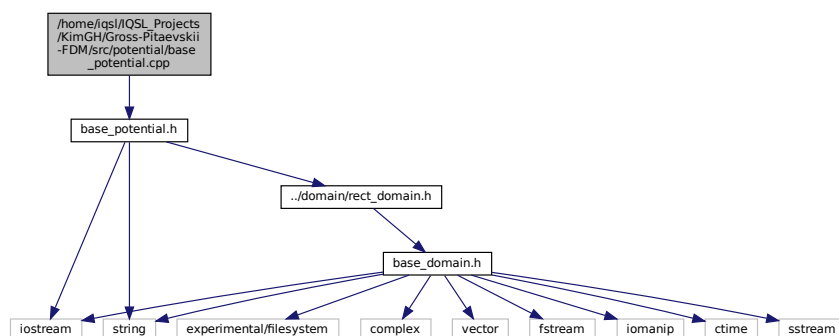
191         if (mpi_use)
192         {
193             // setup MPI if mpi_use
194             gSweeper.set_MPI_info(rank, size);
195         }
196         // Setup whether print through standard output or not
197         gSweeper.set_print_info(print_info);
198         // Setup whether save data or not
199         gSweeper.set_save_data(save_data);
200         // Run sweeping
201         gSweeper.run(domain, initial_condition, potential);
202     }
203     else
204     {
205         std::cerr << "Unexpected initial condition" << std::endl;
206     }
207 }
208 else
209 {
210     std::cerr << "Unexpected initial condition" << std::endl;
211 }
212 }
213 else if (parameters.main_parameters.calculation_type == "anisotropy_sweep")
214 {
215     // Prepare Harmonic Potential Sweeper
216     HPSweeper hpSweeper = HPSweeper(parameters.main_parameters.float_parameters["sweep_start"],
217                                     parameters.main_parameters.float_parameters["sweep_end"],
218                                     parameters.main_parameters.int_parameters["sweep_count"],
219                                     bool(parameters.main_parameters.int_parameters["end_point"]));
220     // Prepare domain
221     RectangularDomain *domain = new RectangularDomain(parameters.domain_parameters.n_x,
222                                                         parameters.domain_parameters.n_y,
223                                                         parameters.domain_parameters.time_start,
224                                                         parameters.domain_parameters.time_end,
225                                                         parameters.domain_parameters.n_time,
226                                                         parameters.domain_parameters.spatial_parameters["x_start"],
227                                                         parameters.domain_parameters.spatial_parameters["x_end"],
228                                                         parameters.domain_parameters.spatial_parameters["y_start"],
229                                                         parameters.domain_parameters.spatial_parameters["y_end"]);
230     // Prepare initial Condition
231     std::function<std::complex<float>(float, float)> initial_cond_function;
232     if (parameters.init_cond_parameters.init_cond_type == "singlegaussian")
233     {
234         // Prepare initial condition
235         auto sigma_x = parameters.init_cond_parameters.init_cond_parameters["sigma_x"];
236         auto sigma_y = parameters.init_cond_parameters.init_cond_parameters["sigma_y"];
237         auto center_x = parameters.init_cond_parameters.init_cond_parameters["center_x"];
238         auto center_y = parameters.init_cond_parameters.init_cond_parameters["center_y"];
239         initial_cond_function = [center_x, center_y, sigma_x, sigma_y](float x, float y)
240         {
241             return std::complex<float>{
242                 float(1.) * expf(-((x - center_x) * (x - center_x) / (sigma_x * sigma_x) + (y -
243 center_y) * (y - center_y) / (sigma_y * sigma_y))));
244             };
245             auto *initial_condition = new InitialCondition(initial_cond_function);
246             initial_condition->assign_to_domain(domain);
247         }
248         if (parameters.equation_parameters.potential_type == "harmonic")
249         {
250             // setup save / MPI use / cuda use / print option
251             float g = parameters.equation_parameters.g;
252             bool save_data = parameters.solver_parameters.save_data;
253             bool print_info = parameters.solver_parameters.print_info;
254             bool cuda_use = (bool)parameters.main_parameters.int_parameters["cuda_use"];
255             bool mpi_use = (bool)parameters.main_parameters.int_parameters["mpi_use"];
256             if (cuda_use)
257             {
258                 hpSweeper.set_CUDA_info(parameters.main_parameters.int_parameters["gpu_count"],
259 parameters.main_parameters.int_parameters["calculation_per_gpu"]);
260             }
261             if (mpi_use)
262             {
263                 hpSweeper.set_MPI_info(rank, size);
264             }
265             hpSweeper.set_print_info(print_info);
266             hpSweeper.set_save_data(save_data);
267             hpSweeper.run(domain, initial_condition, g);
268         }
269     }
270 }

```



```
#include "base_potential.h"
```

Include dependency graph for base_potential.cpp:



5.12.1 Detailed Description

Implementation of the methods in potential class.

Author

Gyeonghun Kim, Minyoung Kim

Version

0.1

Date

2022-06-08

Copyright

Copyright (c) 2022

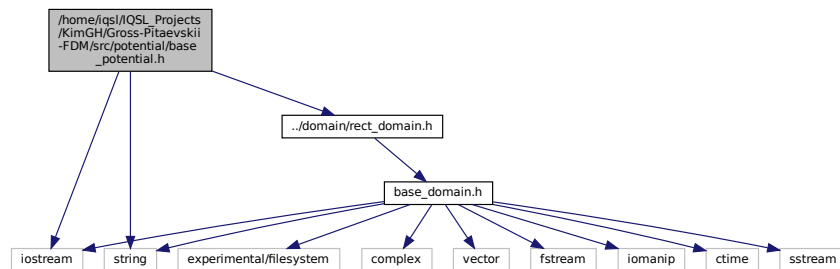
5.13 /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/potential/base_potential.h File Reference

Header for the base potential class. Heritate this class for create custom potential shape.

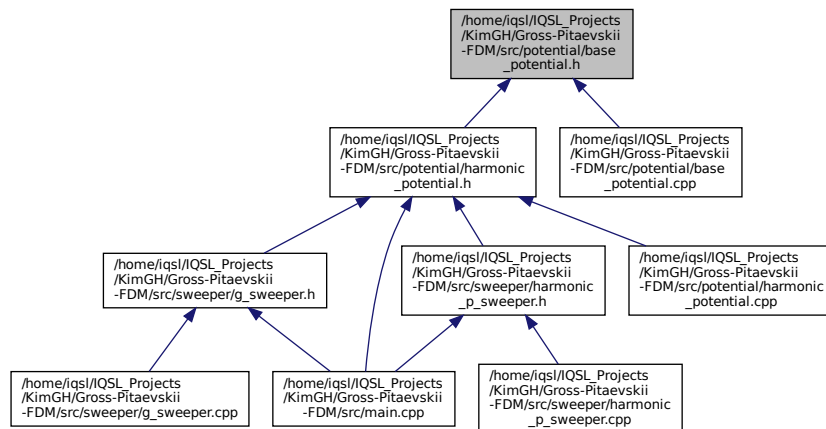
```
#include <iostream>
#include <string>
```

```
#include "../domain/rect_domain.h"
```

Include dependency graph for base_potential.h:



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



Classes

- class [BasePotential](#)

5.13.1 Detailed Description

Header for the base potential class. Heritate this class for create custom potential shape.

Author

Gyeonghun Kim, Minyoung Kim

Version

0.1

Date

2022-06-08

Copyright

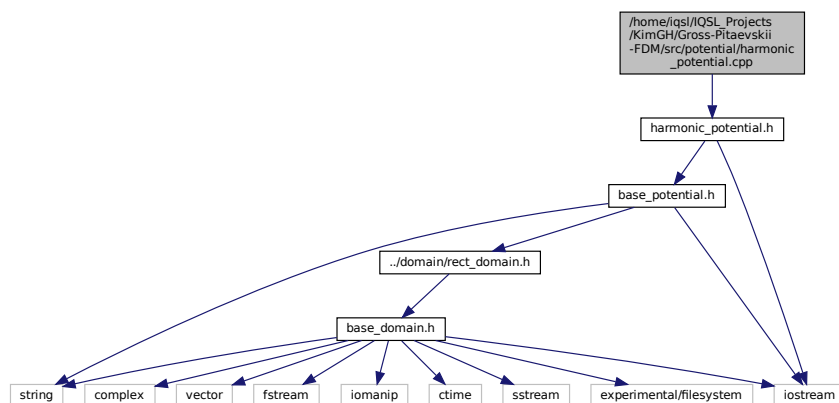
Copyright (c) 2022

5.14 /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/potential/harmonic_potential.cpp File Reference

Implementation of harmonic potential methods.

```
#include "harmonic_potential.h"
```

Include dependency graph for harmonic_potential.cpp:



5.14.1 Detailed Description

Implementation of harmonic potential methods.

Author

Minyoung Kim, Gyeonghun Kim

Version

0.1

Date

2022-06-08

Copyright

Copyright (c) 2022

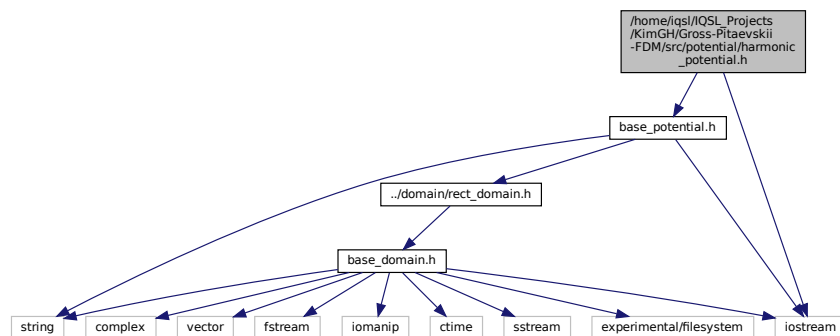
5.15 /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/potential/harmonic_potential.h File Reference

Header file for harmoni potential ($V = 0.5 * (\omega_x^2 * x^2 + \omega_y^2 * y^2)$)

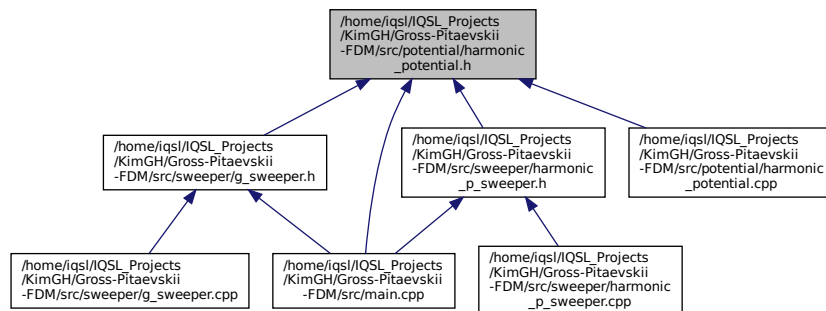
```
#include <iostream>
```

```
#include "base_potential.h"
```

Include dependency graph for harmonic_potential.h:



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



Classes

- class [HarmonicPotential](#)

5.15.1 Detailed Description

Header file for harmoni potential ($V = 0.5 * (\omega_x^2 * x^2 + \omega_y^2 * y^2)$)

Author

Minyoung Kim, Gyeonghun Kim

Version

0.1

Date

2022-06-08

Copyright

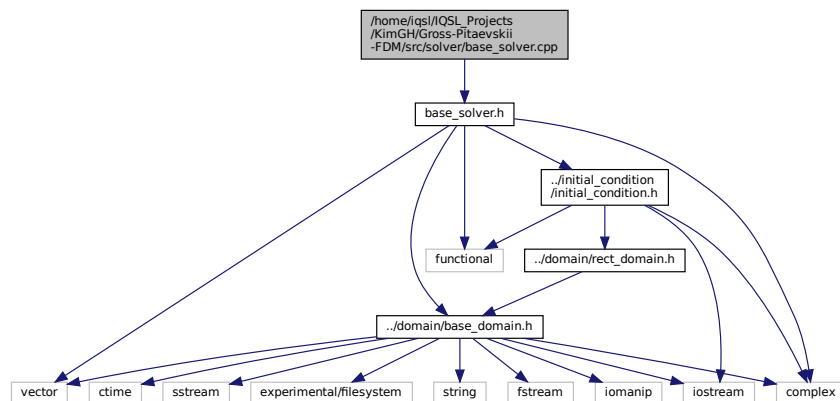
Copyright (c) 2022

5.16 /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/solver/base_solver.cpp File Reference

Header file for base solver.

```
#include "base_solver.h"
```

Include dependency graph for base_solver.cpp:



5.16.1 Detailed Description

Header file for base solver.

Author

Minyoung Kim, Gyeonghun Kim

Version

0.1

Date

2022-06-05

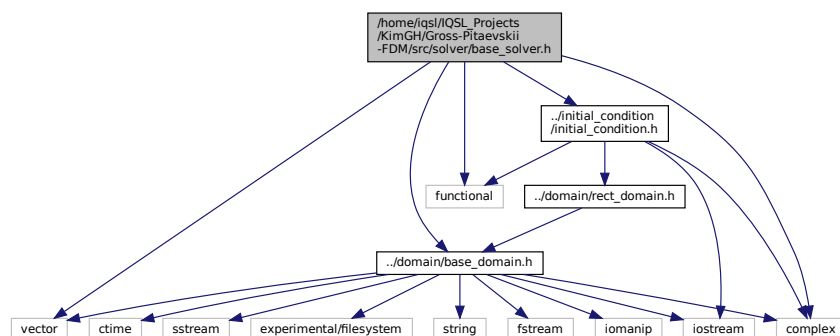
Copyright

Copyright (c) 2022

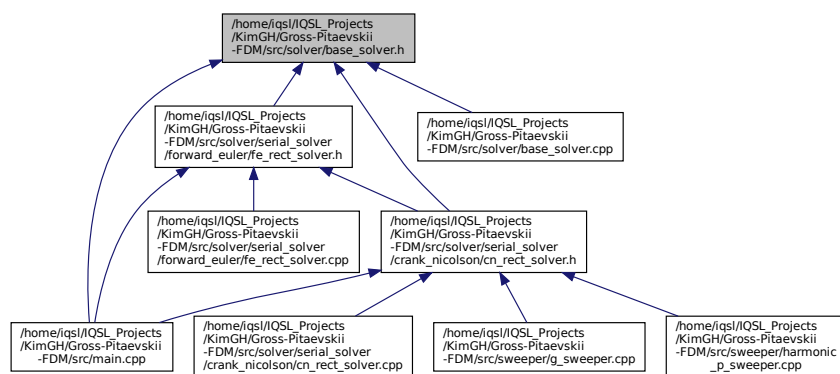
5.17 /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/solver/base_solver.h File Reference

Implementation file for base solver.

```
#include <complex>
#include <vector>
#include <functional>
#include "../domain/base_domain.h"
#include "../initial_condition/initial_condition.h"
Include dependency graph for base_solver.h:
```



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



Classes

- class [BaseSolver](#)

Base Solver for Gross Piteavskill finite difference solver.

5.17.1 Detailed Description

Implementation file for base solver.

Author

Minyoung Kim, Gyeonghun Kim

Version

0.1

Date

2022-06-05

Copyright

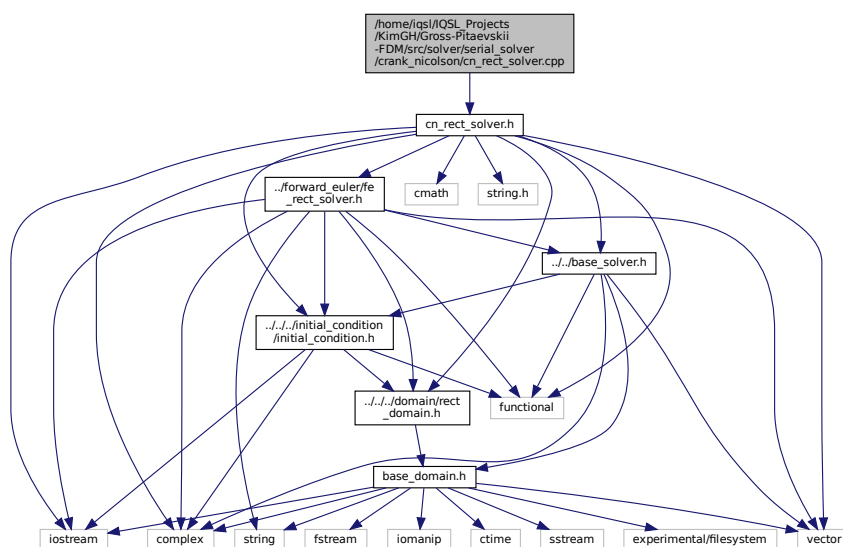
Copyright (c) 2022

5.18 /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FD↩ M/src/solver/serial_solver/crank_nicolson/cn_rect_solver.cpp File Reference

Implementation file for serial crank nicolson solver.

```
#include "cn_rect_solver.h"
```

Include dependency graph for cn_rect_solver.cpp:



5.18.1 Detailed Description

Implementation file for serial crank nicolson solver.

Author

Minyoung Kim, Gyeonghun Kim

Version

0.1

Date

2022-06-05

Copyright

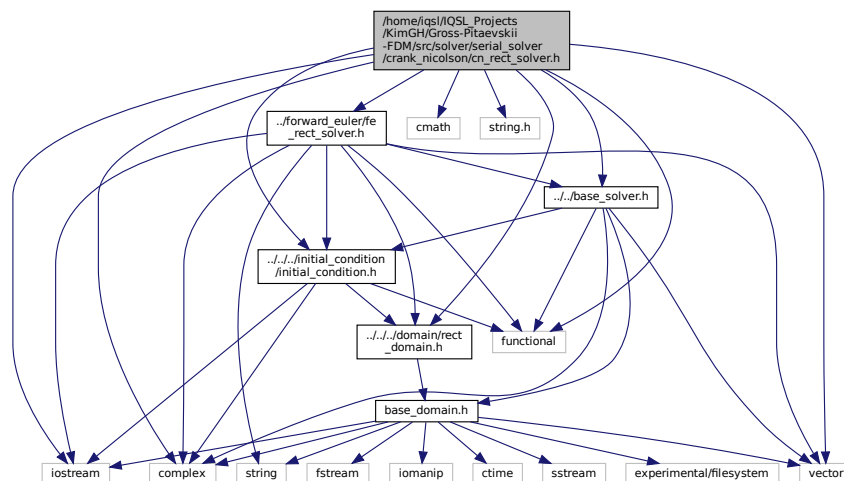
Copyright (c) 2022

5.19 /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/solver/serial_solver/crank_nicolson/cn_rect_solver.h File Reference

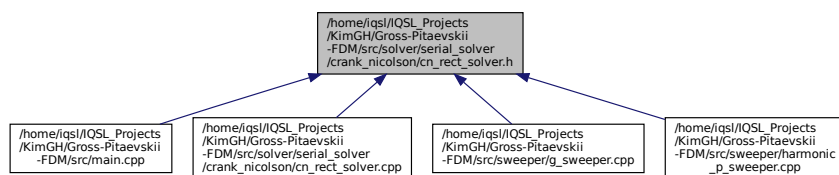
Header file for serial crank nicolson solver.

```
#include <complex>
#include <vector>
#include <functional>
#include <iostream>
#include <cmath>
#include <string.h>
#include "../domain/rect_domain.h"
#include "../initial_condition/initial_condition.h"
#include "../base_solver.h"
#include "../forward_euler/fe_rect_solver.h"
```

Include dependency graph for cn_rect_solver.h:



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



Classes

- class [CNRectSolver](#)

5.19.1 Detailed Description

Header file for serial crank nicolson solver.

Author

Minyoung Kim, Gyeonghun Kim

Version

0.1

Date

2022-06-05

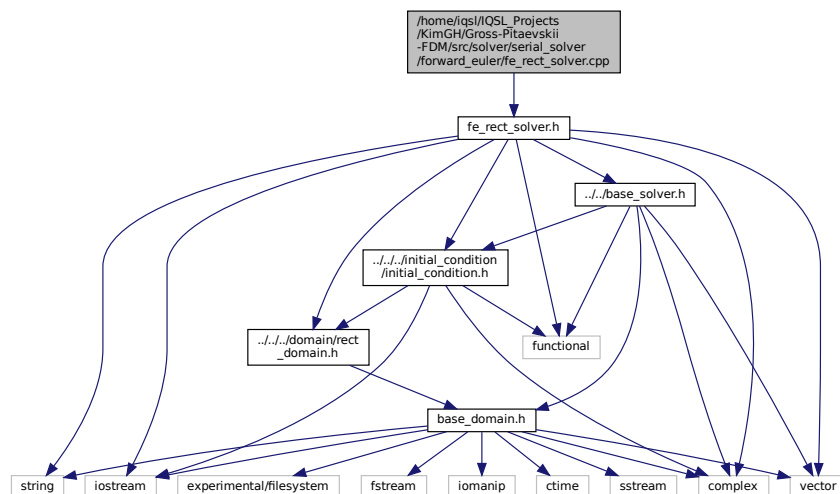
Copyright

Copyright (c) 2022

5.20 /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FD↵ M/src/solver/serial_solver/forward_euler/fe_rect_solver.cpp File Reference

Implementation file for serial forward euler solver methods.

```
#include "fe_rect_solver.h"
Include dependency graph for fe_rect_solver.cpp:
```



5.20.1 Detailed Description

Implementation file for serial forward euler solver methods.

Author

Minyoung Kim, Gyeonghun Kim

Version

0.1

Date

2022-06-05

Copyright

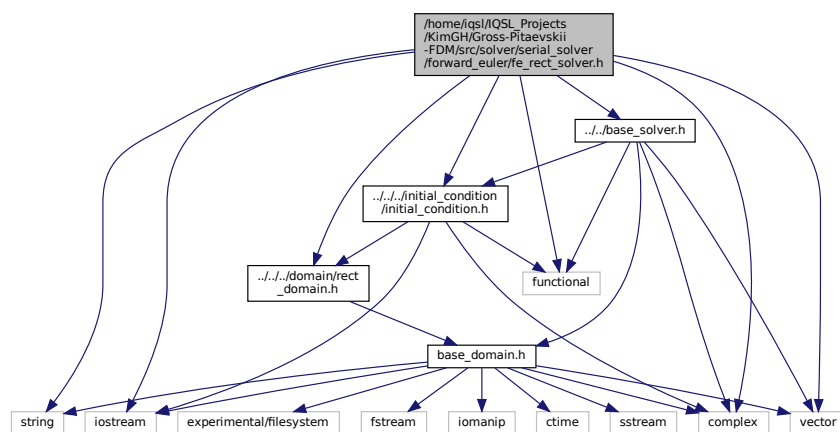
Copyright (c) 2022

5.21 /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/solver/serial_solver/forward_euler/fe_rect_solver.h File Reference

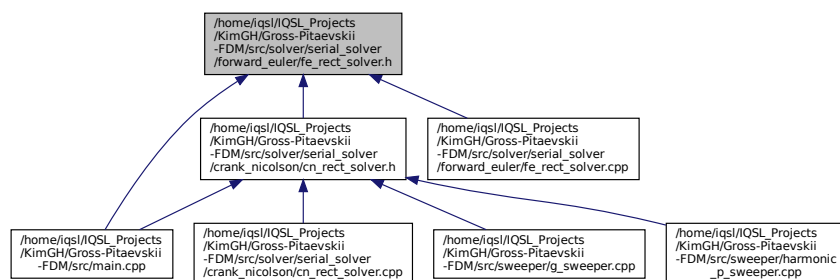
Header file for serial forward euler solver.

```
#include <complex>
#include <vector>
#include <functional>
#include <iostream>
#include <string>
#include "../domain/rect_domain.h"
#include "../initial_condition/initial_condition.h"
#include "../base_solver.h"
```

Include dependency graph for `fe_rect_solver.h`:



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



Classes

- class [FERectSolver](#)
Forward Euler Serial Solver.

5.21.1 Detailed Description

Header file for serial forward euler solver.

Author

Minyoung Kim, Gyeonghun Kim

Version

0.1

Date

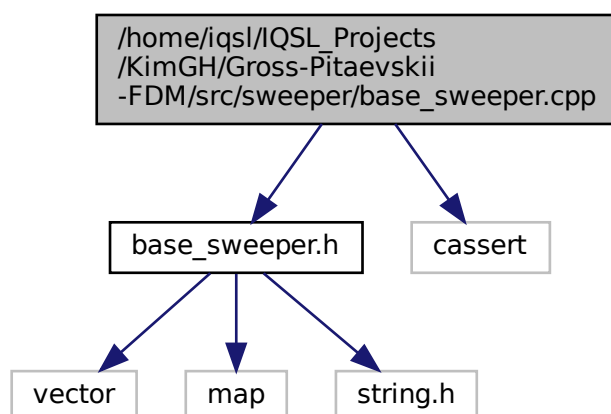
2022-06-05

Copyright

Copyright (c) 2022

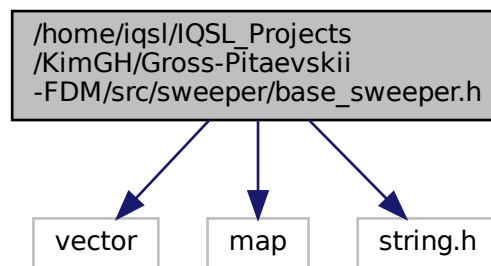
5.22 /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FD↵ M/src/sweeper/base_sweeper.cpp File Reference

```
#include "base_sweeper.h"  
#include <cassert>  
Include dependency graph for base_sweeper.cpp:
```

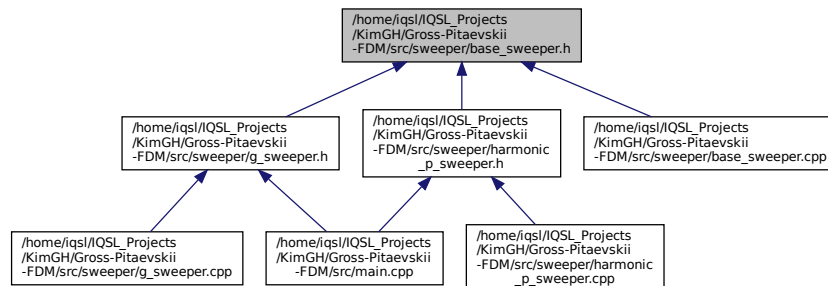


5.23 /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FD↵ M/src/sweeper/base_sweeper.h File Reference

```
#include <vector>
#include <map>
#include <string.h>
Include dependency graph for base_sweeper.h:
```



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



Classes

- class [BaseSweeper](#)

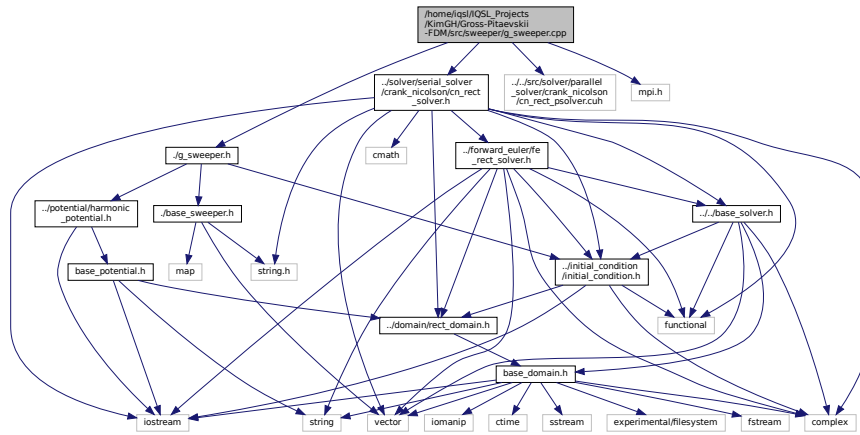
5.24 /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FD↵ M/src/sweeper/g_sweeper.cpp File Reference

```
#include "../g_sweeper.h"
#include "../../solver/serial_solver/crank_nicolson/cn_rect_solver.h"
```

```
#include "../src/solver/parallel_solver/crank_nicolson/cn_rect_psolver.cuh"
```

```
#include <mpi.h>
```

Include dependency graph for g_sweeper.cpp:



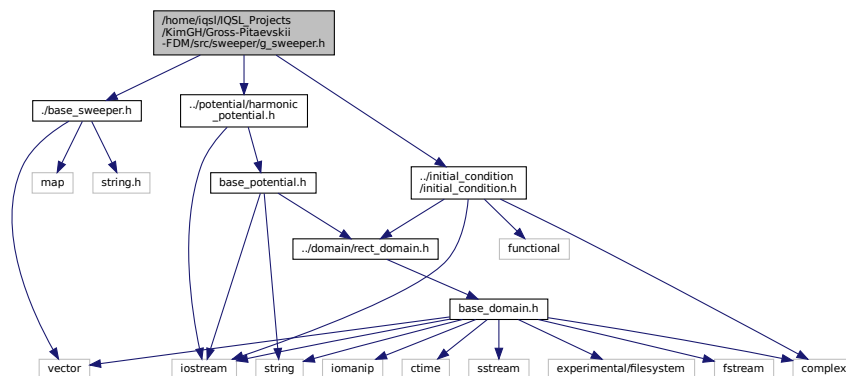
5.25 /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/sweeper/g_sweeper.h File Reference

```
#include "../base_sweeper.h"
```

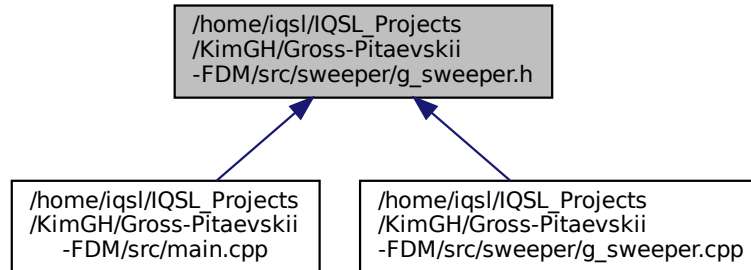
```
#include "../potential/harmonic_potential.h"
```

```
#include "../initial_condition/initial_condition.h"
```

Include dependency graph for g_sweeper.h:



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



Classes

- class [GSweeper](#)

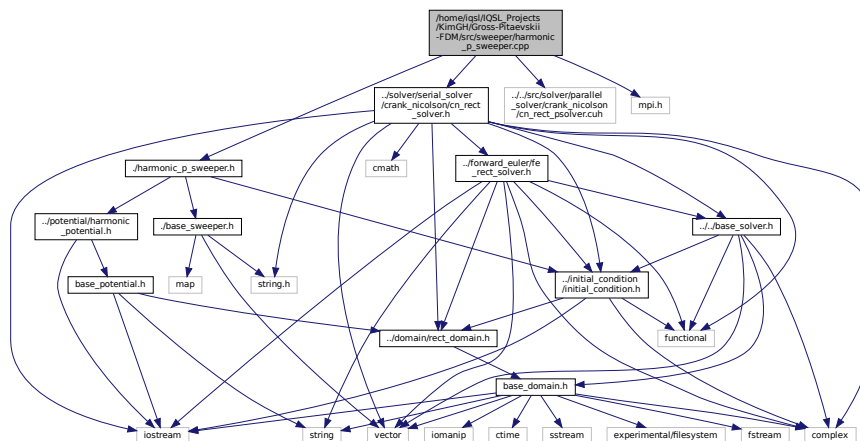
5.26 /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/sweeper/harmonic_p_sweeper.cpp File Reference

```

#include "../harmonic_p_sweeper.h"
#include "../../solver/serial_solver/crank_nicolson/cn_rect_solver.h"
#include "../../../src/solver/parallel_solver/crank_nicolson/cn_rect_psolver.cuh"
#include <mpi.h>

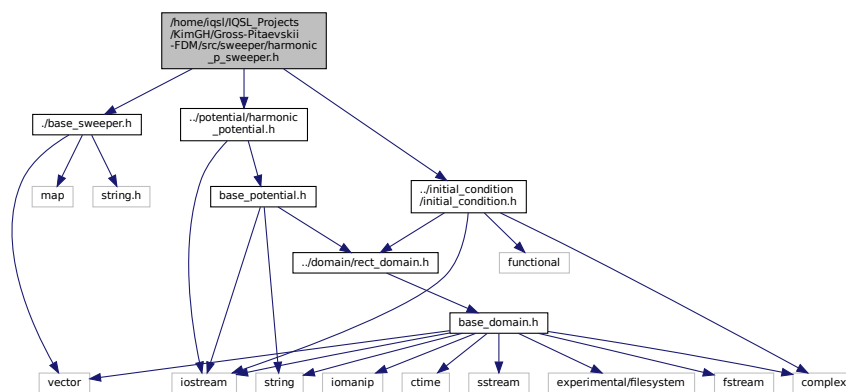
```

Include dependency graph for harmonic_p_sweeper.cpp:

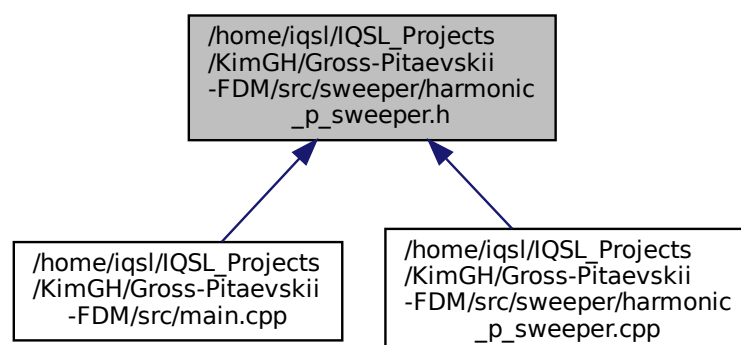


5.27 /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/sweeper/harmonic_p_sweeper.h File Reference

```
#include "../base_sweeper.h"
#include "../../potential/harmonic_potential.h"
#include "../../initial_condition/initial_condition.h"
Include dependency graph for harmonic_p_sweeper.h:
```



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

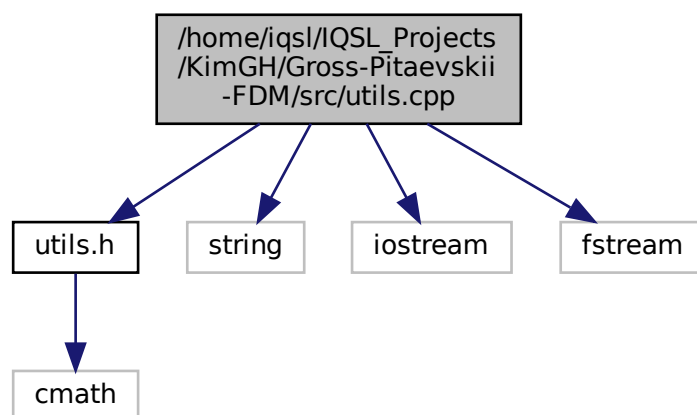


Classes

- class [HPSweeper](#)

5.28 /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/Utils.cpp File Reference

```
#include "utils.h"
#include <string>
#include <iostream>
#include <fstream>
Include dependency graph for utils.cpp:
```



Functions

- bool `is_close` (float `a`, float `b`, float `tolerance`)

5.28.1 Function Documentation

5.28.1.1 `is_close()`

```
bool is_close (
    float a,
    float b,
    float tolerance )
```

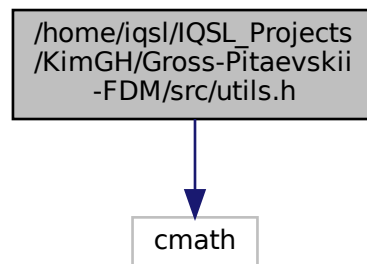
Definition at line 5 of file `utils.cpp`.

```
5 {
6     return std::abs(a - b) < tolerance;
7 }
```

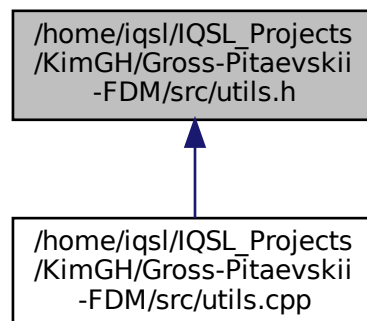
5.29 /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/utils.h File Reference

```
#include <cmath>
```

Include dependency graph for utils.h:



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



Functions

- bool `is_close` (float a, float b, float tolerance)

5.29.1 Function Documentation

5.29.1.1 is_close()

```
bool is_close (
    float a,
    float b,
    float tolerance )
```

Definition at line 5 of file utils.cpp.

```
5 {
6     return std::abs(a - b) < tolerance;
7 }
```


Index

- /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/CMakeLists.txt, [151](#)
- /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/configuration/config_parser.cpp, [152](#)
- /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/configuration/config_parser.h, [153](#)
- /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/configuration/parameters.h, [154](#)
- /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/domain/base_domain.cpp, [156](#)
- /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/domain/base_domain.h, [157](#)
- /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/domain/rect_domain.cpp, [158](#)
- /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/domain/rect_domain.h, [159](#)
- /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/initial_condition/initial_condition.cpp, [160](#)
- /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/initial_condition/initial_condition.h, [161](#)
- /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/main.cpp, [163](#)
- /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/potential/base_potential.cpp, [167](#)
- /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/potential/base_potential.h, [168](#)
- /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/potential/harmonic_potential.cpp, [170](#)
- /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/potential/harmonic_potential.h, [171](#)
- /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/solver/base_solver.cpp, [172](#)
- /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/solver/base_solver.h, [173](#)
- /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/solver/serial_solver/krank_nicolson/cn_rect_solver.cpp, [174](#)
- /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/solver/serial_solver/krank_nicolson/cn_rect_solver.h, [175](#)
- /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/solver/serial_solver/forward_euler/fe_rect_solver.cpp, [176](#)
- /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/solver/serial_solver/forward_euler/fe_rect_solver.h, [177](#)
- /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/solver/base_sweeper.cpp, [179](#)
- /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/solver/base_sweeper.h, [180](#)
- /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/solver/g_sweeper.cpp, [180](#)
- /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/solver/g_sweeper.h, [181](#)
- /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/solver/harmonic_p_sweeper.cpp, [182](#)
- /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/solver/harmonic_p_sweeper.h, [183](#)
- /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/utls.cpp, [184](#)
- /home/iqsl/IQSL_Projects/KimGH/Gross-Pitaevskii-FDM/src/utls.h, [185](#)
- ~BaseDomain
 - BaseDomain, [11](#)
- ~BaseSolver
 - BaseSolver, [30](#)
- ~BaseSpatialGrid
 - BaseSpatialGrid, [34](#)
- ~BaseSweeper
 - BaseSweeper, [41](#)
- ~ConfigParser
 - ConfigParser, [64](#)
- ~GridPoint
 - GridPoint, [85](#)
- ~RectangularDomain
 - RectangularDomain, [125](#)
- ~RectangularSpatialGrid
 - RectangularSpatialGrid, [144](#)
- assign_initial_value
 - BaseDomain, [11](#)
 - RectangularDomain, [125](#)
- assign_to_domain
 - RectangularDomain, [115](#)
- assign_wave_function
 - BaseDomain, [12](#)
 - RectangularDomain, [126](#)
- at
 - BaseDomain, [13](#)
 - BaseSpatialGrid, [34](#)
 - RectangularDomain, [126](#)
 - RectangularSpatialGrid, [144](#)
- BaseDomain, [7](#)

- ~BaseDomain, 11
- assign_initial_value, 11
- assign_wave_function, 12
- at, 13
- BaseDomain, 10, 11
- current_grid, 22
- current_time_index, 22
- dt, 22
- generate_directory_name, 13
- generate_single_txt_file, 14
- get_current_time_index, 15
- get_dt, 15
- get_infinitesimal_distance1, 16
- get_infinitesimal_distance2, 16
- get_null_gridpt, 17
- get_num_grid_1, 17
- get_num_grid_2, 18
- get_num_times, 18
- get_path, 19
- get_t_end, 19
- get_t_start, 19
- normalize, 19
- null_gridpt, 22
- num_grid_1, 22
- num_grid_2, 22
- num_times, 23
- old_grid, 23
- PATH, 23
- print_directory_info, 20
- reset, 20
- t_end, 23
- t_start, 23
- time_at, 21
- update_time, 21
- BasePotential, 24
 - BasePotential, 25
 - calcualte_potential_in_grid, 25
 - get_name, 26
 - name, 27
 - potential_function, 26
- BaseSolver, 28
 - ~BaseSolver, 30
 - BaseSolver, 29, 30
 - g, 31
 - string_info, 31
 - temporal_equation, 30
- BaseSpatialGrid, 31
 - ~BaseSpatialGrid, 34
 - at, 34
 - BaseSpatialGrid, 33, 34
 - get_infinitesimal_distance1, 35
 - get_infinitesimal_distance2, 35
 - infinitesimal_distance_1, 37
 - infinitesimal_distance_2, 37
 - normalize, 36
 - num_grid_1, 37
 - num_grid_2, 37
 - spatial_data, 37
- BaseSweeper, 38
 - ~BaseSweeper, 41
 - BaseSweeper, 41
 - CUDA_use, 45
 - end, 45
 - endpoint, 46
 - generate_num_list, 42
 - get_end, 42
 - get_number_of_pts, 42
 - get_start, 43
 - get_value_from_idx, 43
 - gpu_limit, 46
 - gpu_num, 46
 - MPI_use, 46
 - num, 46
 - num_list, 46
 - print_info, 47
 - rank, 47
 - save_data, 47
 - set_CUDA_info, 43
 - set_MPI_info, 44
 - set_print_info, 44
 - set_save_data, 45
 - size, 47
 - start, 47
- calcualte_potential_in_grid
 - BasePotential, 25
 - HarmonicPotential, 100
- calculate_error
 - CNRectSolver, 51
- calculation_type
 - MainParameters, 118
- cmake_minimum_required
 - CMakeLists.txt, 151
- CMakeLists.txt
 - cmake_minimum_required, 151
 - target_link_libraries, 151
- CNRectSolver, 48
 - calculate_error, 51
 - CNRectSolver, 50
 - domain, 62
 - g, 62
 - get_potential_value, 52
 - guess, 63
 - initialize_guess_with_forward_euler, 53
 - solve, 54, 56
 - solve_single_time, 57, 58
 - string_info, 63
 - temporal_equation, 59
 - update_guess, 61
- config_name
 - Parameters, 119
- ConfigParser, 63
 - ~ConfigParser, 64
 - ConfigParser, 64
 - get_default, 64
 - parse, 65
- CUDA_use

- BaseSweeper, 45
- GSweeper, 95
- HPSweeper, 111
- current_grid
 - BaseDomain, 22
 - RectangularDomain, 137
- current_time_index
 - BaseDomain, 22
 - RectangularDomain, 137
- domain
 - CNRectSolver, 62
 - FERectSolver, 83
- domain_parameters
 - Parameters, 119
- domain_type
 - DomainParameters, 72
- DomainParameters, 72
 - domain_type, 72
 - n_time, 73
 - n_x, 73
 - n_y, 73
 - spatial_parameters, 73
 - time_end, 73
 - time_start, 73
- dt
 - BaseDomain, 22
 - RectangularDomain, 137
- end
 - BaseSweeper, 45
 - GSweeper, 95
 - HPSweeper, 111
- endpoint
 - BaseSweeper, 46
 - GSweeper, 96
 - HPSweeper, 112
- equation_parameters
 - Parameters, 120
- EquationParameters, 74
 - g, 74
 - potential_parameters, 75
 - potential_type, 75
- FERectSolver, 75
 - domain, 83
 - FERectSolver, 78
 - g, 83
 - get_potential_value, 79
 - solve, 79
 - solve_single_time, 81
 - string_info, 84
 - temporal_equation, 82
- float_parameters
 - MainParameters, 118
- g
 - BaseSolver, 31
 - CNRectSolver, 62
 - EquationParameters, 74
 - FERectSolver, 83
- generate_directory_name
 - BaseDomain, 13
 - RectangularDomain, 127
- generate_num_list
 - BaseSweeper, 42
 - GSweeper, 90
 - HPSweeper, 106
- generate_single_txt_file
 - BaseDomain, 14
 - RectangularDomain, 128
- get_current_time_index
 - BaseDomain, 15
 - RectangularDomain, 129
- get_default
 - ConfigParser, 64
- get_dt
 - BaseDomain, 15
 - RectangularDomain, 129
- get_end
 - BaseSweeper, 42
 - GSweeper, 90
 - HPSweeper, 106
- get_infinitesimal_distance1
 - BaseDomain, 16
 - BaseSpatialGrid, 35
 - RectangularDomain, 130
 - RectangularSpatialGrid, 145
- get_infinitesimal_distance2
 - BaseDomain, 16
 - BaseSpatialGrid, 35
 - RectangularDomain, 130
 - RectangularSpatialGrid, 145
- get_name
 - BasePotential, 26
 - HarmonicPotential, 101
- get_null_gridpt
 - BaseDomain, 17
 - RectangularDomain, 131
- get_num_grid_1
 - BaseDomain, 17
 - RectangularDomain, 131
- get_num_grid_2
 - BaseDomain, 18
 - RectangularDomain, 132
- get_num_times
 - BaseDomain, 18
 - RectangularDomain, 132
- get_number_of_pts
 - BaseSweeper, 42
 - GSweeper, 90
 - HPSweeper, 106
- get_path
 - BaseDomain, 19
 - RectangularDomain, 133
- get_potential_value
 - CNRectSolver, 52

- FERectSolver, 79
- get_start
 - BaseSweeper, 43
 - GSweeper, 90
 - HPSweeper, 106
- get_t_end
 - BaseDomain, 19
 - RectangularDomain, 133
- get_t_start
 - BaseDomain, 19
 - RectangularDomain, 133
- get_value_from_idx
 - BaseSweeper, 43
 - GSweeper, 91
 - HPSweeper, 107
- get_x_end
 - RectangularDomain, 133
- get_x_start
 - RectangularDomain, 134
- get_y_end
 - RectangularDomain, 134
- get_y_start
 - RectangularDomain, 134
- gpu_limit
 - BaseSweeper, 46
 - GSweeper, 96
 - HPSweeper, 112
- gpu_num
 - BaseSweeper, 46
 - GSweeper, 96
 - HPSweeper, 112
- GridPoint, 84
 - ~GridPoint, 85
 - GridPoint, 85
 - value, 85
 - x, 86
 - y, 86
- GSweeper, 86
 - CUDA_use, 95
 - end, 95
 - endpoint, 96
 - generate_num_list, 90
 - get_end, 90
 - get_number_of_pts, 90
 - get_start, 90
 - get_value_from_idx, 91
 - gpu_limit, 96
 - gpu_num, 96
 - GSweeper, 89
 - MPI_use, 96
 - num, 96
 - num_list, 96
 - print_info, 97
 - rank, 97
 - run, 91
 - save_data, 97
 - set_CUDA_info, 93
 - set_MPI_info, 94
 - set_print_info, 94
 - set_save_data, 95
 - size, 97
 - start, 97
- guess
 - CNRectSolver, 63
- HarmonicPotential, 98
 - calcualte_potential_in_grid, 100
 - get_name, 101
 - HarmonicPotential, 100
 - name, 102
 - omega_x, 102
 - omega_y, 102
 - potential_function, 101
- HPSweeper, 103
 - CUDA_use, 111
 - end, 111
 - endpoint, 112
 - generate_num_list, 106
 - get_end, 106
 - get_number_of_pts, 106
 - get_start, 106
 - get_value_from_idx, 107
 - gpu_limit, 112
 - gpu_num, 112
 - HPSweeper, 105
 - MPI_use, 112
 - num, 112
 - num_list, 112
 - print_info, 113
 - rank, 113
 - run, 107
 - save_data, 113
 - set_CUDA_info, 109
 - set_MPI_info, 110
 - set_print_info, 110
 - set_save_data, 111
 - size, 113
 - start, 113
- infinitesimal_distance_1
 - BaseSpatialGrid, 37
 - RectangularSpatialGrid, 146
- infinitesimal_distance_2
 - BaseSpatialGrid, 37
 - RectangularSpatialGrid, 146
- init_cond_parameters
 - InitialConditionParameters, 117
 - Parameters, 120
- init_cond_type
 - InitialConditionParameters, 117
- initial_condition_function
 - InitialCondition, 116
- InitialCondition, 114
 - assign_to_domain, 115
 - initial_condition_function, 116
 - InitialCondition, 114
- InitialConditionParameters, 116

- init_cond_parameters, 117
 - init_cond_type, 117
- initialize_guess_with_forward_euler
 - CNRectSolver, 53
- int_parameters
 - MainParameters, 118
 - SolverParameters, 149
- is_close
 - utils.cpp, 184
 - utils.h, 185
- main
 - main.cpp, 163
- main.cpp
 - main, 163
- main_parameters
 - Parameters, 120
- MainParameters, 117
 - calculation_type, 118
 - float_parameters, 118
 - int_parameters, 118
- method
 - SolverParameters, 149
- MPI_use
 - BaseSweeper, 46
 - GSweeper, 96
 - HPSweeper, 112
- n_time
 - DomainParameters, 73
- n_x
 - DomainParameters, 73
- n_y
 - DomainParameters, 73
- name
 - BasePotential, 27
 - HarmonicPotential, 102
- normalize
 - BaseDomain, 19
 - BaseSpatialGrid, 36
 - RectangularDomain, 134
 - RectangularSpatialGrid, 145
- null_gridpt
 - BaseDomain, 22
 - RectangularDomain, 138
- num
 - BaseSweeper, 46
 - GSweeper, 96
 - HPSweeper, 112
- num_grid_1
 - BaseDomain, 22
 - BaseSpatialGrid, 37
 - RectangularDomain, 138
 - RectangularSpatialGrid, 146
- num_grid_2
 - BaseDomain, 22
 - BaseSpatialGrid, 37
 - RectangularDomain, 138
 - RectangularSpatialGrid, 147
- num_list
 - BaseSweeper, 46
 - GSweeper, 96
 - HPSweeper, 112
- num_times
 - BaseDomain, 23
 - RectangularDomain, 138
- old_grid
 - BaseDomain, 23
 - RectangularDomain, 138
- omega_x
 - HarmonicPotential, 102
- omega_y
 - HarmonicPotential, 102
- Parameters, 119
 - config_name, 119
 - domain_parameters, 119
 - equation_parameters, 120
 - init_cond_parameters, 120
 - main_parameters, 120
 - solver_parameters, 120
- parse
 - ConfigParser, 65
- PATH
 - BaseDomain, 23
 - RectangularDomain, 138
- potential_function
 - BasePotential, 26
 - HarmonicPotential, 101
- potential_grid
 - RectangularDomain, 139
- potential_parameters
 - EquationParameters, 75
- potential_type
 - EquationParameters, 75
- print_directory_info
 - BaseDomain, 20
 - RectangularDomain, 135
- print_info
 - BaseSweeper, 47
 - GSweeper, 97
 - HPSweeper, 113
 - SolverParameters, 149
- rank
 - BaseSweeper, 47
 - GSweeper, 97
 - HPSweeper, 113
- RectangularDomain, 121
 - ~RectangularDomain, 125
 - assign_initial_value, 125
 - assign_wave_function, 126
 - at, 126
 - current_grid, 137
 - current_time_index, 137
 - dt, 137
 - generate_directory_name, 127

- generate_single_txt_file, 128
- get_current_time_index, 129
- get_dt, 129
- get_infinitesimal_distance1, 130
- get_infinitesimal_distance2, 130
- get_null_gridpt, 131
- get_num_grid_1, 131
- get_num_grid_2, 132
- get_num_times, 132
- get_path, 133
- get_t_end, 133
- get_t_start, 133
- get_x_end, 133
- get_x_start, 134
- get_y_end, 134
- get_y_start, 134
- normalize, 134
- null_gridpt, 138
- num_grid_1, 138
- num_grid_2, 138
- num_times, 138
- old_grid, 138
- PATH, 138
- potential_grid, 139
- print_directory_info, 135
- RectangularDomain, 124
- reset, 135
- t_end, 139
- t_start, 139
- time_at, 136
- update_time, 136
- x_end, 139
- x_start, 139
- y_end, 139
- y_start, 140
- RectangularSpatialGrid, 140
 - ~RectangularSpatialGrid, 144
 - at, 144
 - get_infinitesimal_distance1, 145
 - get_infinitesimal_distance2, 145
 - infinitesimal_distance_1, 146
 - infinitesimal_distance_2, 146
 - normalize, 145
 - num_grid_1, 146
 - num_grid_2, 147
 - RectangularSpatialGrid, 143
 - spatial_data, 147
 - x_end, 147
 - x_start, 147
 - y_end, 147
 - y_start, 147
- reset
 - BaseDomain, 20
 - RectangularDomain, 135
- run
 - GSweeper, 91
 - HPSweeper, 107
- run_parallel
 - SolverParameters, 149
- save_data
 - BaseSweeper, 47
 - GSweeper, 97
 - HPSweeper, 113
 - SolverParameters, 149
- set_CUDA_info
 - BaseSweeper, 43
 - GSweeper, 93
 - HPSweeper, 109
- set_MPI_info
 - BaseSweeper, 44
 - GSweeper, 94
 - HPSweeper, 110
- set_print_info
 - BaseSweeper, 44
 - GSweeper, 94
 - HPSweeper, 110
- set_save_data
 - BaseSweeper, 45
 - GSweeper, 95
 - HPSweeper, 111
- size
 - BaseSweeper, 47
 - GSweeper, 97
 - HPSweeper, 113
- solve
 - CNRectSolver, 54, 56
 - FERectSolver, 79
- solve_single_time
 - CNRectSolver, 57, 58
 - FERectSolver, 81
- solver_parameters
 - Parameters, 120
 - SolverParameters, 149
- SolverParameters, 148
 - int_parameters, 149
 - method, 149
 - print_info, 149
 - run_parallel, 149
 - save_data, 149
 - solver_parameters, 149
- spatial_data
 - BaseSpatialGrid, 37
 - RectangularSpatialGrid, 147
- spatial_parameters
 - DomainParameters, 73
- start
 - BaseSweeper, 47
 - GSweeper, 97
 - HPSweeper, 113
- string_info
 - BaseSolver, 31
 - CNRectSolver, 63
 - FERectSolver, 84
- t_end
 - BaseDomain, 23

- RectangularDomain, [139](#)
- t_start
 - BaseDomain, [23](#)
 - RectangularDomain, [139](#)
- target_link_libraries
 - CMakeLists.txt, [151](#)
- temporal_equation
 - BaseSolver, [30](#)
 - CNRectSolver, [59](#)
 - FERectSolver, [82](#)
- time_at
 - BaseDomain, [21](#)
 - RectangularDomain, [136](#)
- time_end
 - DomainParameters, [73](#)
- time_start
 - DomainParameters, [73](#)
- update_guess
 - CNRectSolver, [61](#)
- update_time
 - BaseDomain, [21](#)
 - RectangularDomain, [136](#)
- utils.cpp
 - is_close, [184](#)
- utils.h
 - is_close, [185](#)
- value
 - GridPoint, [85](#)
- x
 - GridPoint, [86](#)
- x_end
 - RectangularDomain, [139](#)
 - RectangularSpatialGrid, [147](#)
- x_start
 - RectangularDomain, [139](#)
 - RectangularSpatialGrid, [147](#)
- y
 - GridPoint, [86](#)
- y_end
 - RectangularDomain, [139](#)
 - RectangularSpatialGrid, [147](#)
- y_start
 - RectangularDomain, [140](#)
 - RectangularSpatialGrid, [147](#)