Optim_VSTRAP

Generated by Doxygen 1.8.13

Contents

Chapter 1

LARA - _A Monte Car∗∗L∗∗o framework for optim∗∗A∗∗l cont∗∗R∗∗ol of plasm∗∗A∗∗_

The program solves optimal control problems governed by the non-linear kinetic equations including external forces and a collision term in a Monte Carlo framwork.

Dependencies and required libraries

The code was optimized for Ubuntu 18.04 LTS. Before downloading the dependencies, make sure that Ubuntu is up-to-date using `sudo apt-get update` and `sudo apt-get upgrade`.

To use the optimizer, `vstrap` must be installed on the machine.

Before compiling the code the following dependencies and libraries must be installed:

- `Armadillo` (this includes lapacke and blas): install using

  ~~~
  + openMP: install using
  ```sudo apt install libomp-de
  ~~~

- boost: install using

  ~~~
  + build-essentials: install using
  + ```sudo apt-get install build-essentia
  ~~~

- cmake: install using

  ~~~
  For **optional** postprocessing ```python3``` should be installed including the packages
  + argparse
  + pyplot from matplotlib
  + tikzplotlib
  + numpy
  + math
  + pandas

  The packages can be installed using the following commands
  ```sh
  sudo apt install python3-pip -y
  pip3 install matplotlib tikzplotlib pandas numpy
  ~~~

Problem specifications

In the file *Optim_input.xml* it is possible to specify the parameters used by the program. View the comments in te file to get information about the purpose of each parameter. The file `src/controller/optim_controller.cpp` is the core of the optimization.

Structure of the code

The source code is structured in five categories:

- **src/controller**: contains auxiliary subroutines like generating of probability density functions (pdf) and controller for input/output
- **src/io**: contains methods for solving the linear kinetic and adjoint linear kinetic problem
- **src/logger**: core of optimization methods; contains important ncg subroutines and armijo-linesearch as well as functions providing the value of the functional and building the gradient
- **src/objects**: contains python files for visualizing the results of the program
- **src/optimization**:

The program has four more plugins:

- **data**: Here, several test-cases are specified
- **optim-vstrap-toolset**: Imporant plugin for the connection between vstrap and the optimizer
- **pprc**: Files for post-processing (python)
- **test**: gtest files

Compile and run the program

After speficying the parameters, it is possible to compile the code and start the program with the following commands executed in the directory containing the `MOCOKI` folder.

```
mkdir build-Optim && cd build-Optim
cmake ../Optim_VSTRAP
make
./Optim_VSTRAP_CMAKE <path/to/>Optim_input.xml
```

Post-processing

There are `python` files to visualize the results of the MOCOKI code. Assuming the build directory `build-Optim` is at the same directory level as the `Optim_VSTRAP` folder, the following commands executed from the `pprc` folder can be used to visualize data generated by the code.

The following command takes files containing data about development of the value of functional, norm of gradient, norm of control and stepsize during the optimization process and plots these.

```
python3 visualize_control.py ../../data/box_shifting_CSSC/interpolated_control_field.xml
       ../../data/global/box_coarse_512.xml
```

The following command gives plots the control in the current iteration. One has to call the functional specifying the current control and the discretization of the physical domain.

```
python3 post_processing_convergence.py ../../../build-Optim/src/results/
```

Using the dockerized version - UNDER CONSTRUCTION

It is possible to install a docker containing all the needed libraries and dependencies using the following commands executed in the folder in which the `mocoki-image.tar` file is located.

```
sudo docker load < mocoki-image.tar #loads image
sudo docker run --name container_mocoki -it mocoki #creates and starts container named 'container_mocoki'
       using the mocoki:latest image
```

After exiting the container, it can be started again using the command

~~~
sudo docker container start -ai container_mocoki

Inside the container go inside the ```MOCOKI``` folder and run
```sh
sh setup_cmake.sh
~~~

This will execute the current version of the code.

You can also change the code outside the docker container and copy it into and from the container using the commands

```
sudo docker cp MOCOKI/ container_mocoki:MOCOKI #copy inside the container
sudo docker cp container_mocoki:build-MOCOKI build-MOCOKI-v08 #copy build folder from container to local
       machine
```

The `sudo` command may be discarded inside docker.

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

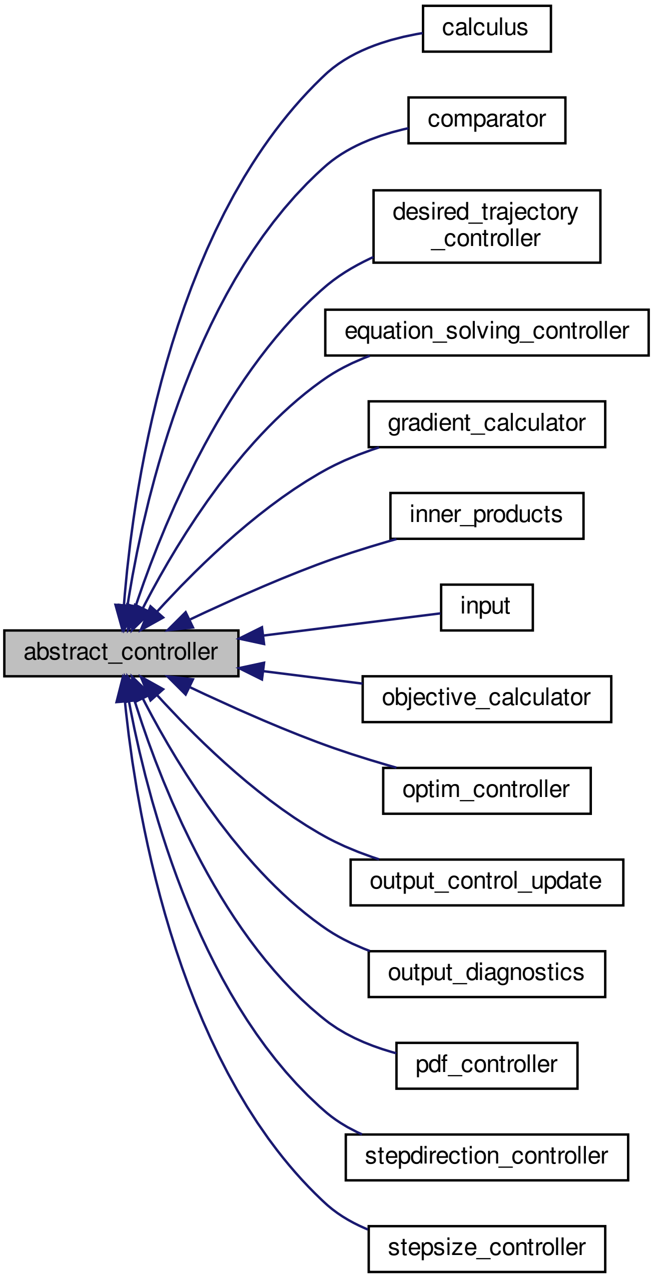
Class Documentation

4.1 abstract_controller Class Reference

The abstract_controller class is inherited by all controller classes.

```
#include <abstract_controller.h>
```

Inheritance diagram for abstract_controller:



Public Member Functions

- data_provider **getData_provider_optim** () const
- void **setData_provider_optim** (const data_provider &value)

4.1.1 Detailed Description

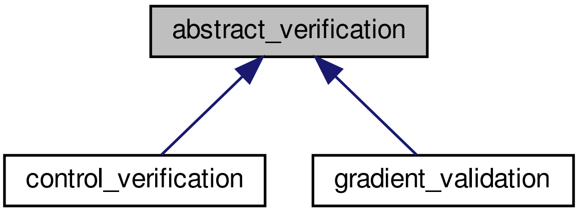
The abstract_controller class is inherited by all controller classes.

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

- /home/jan/Promotion_linuxPC/Optim_VSTRAP/src/controller/abstract_controller.h
- /home/jan/Promotion_linuxPC/Optim_VSTRAP/src/controller/abstract_controller.cpp

4.2 abstract_verification Class Reference

Inheritance diagram for abstract_verification:



Public Member Functions

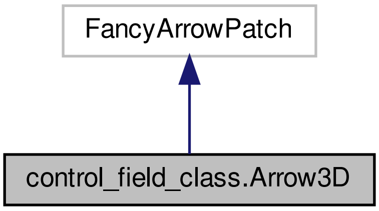
- data_provider **getData_provider_validation** () const
- void **setData_provider_validation** (const data_provider &value)
- data_provider **getData_provider_optim** () const
- void **setData_provider_optim** (const data_provider &value)

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

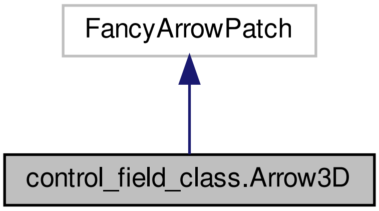
- /home/jan/Promotion_linuxPC/Optim_VSTRAP/vldn/controller/abstract_validation.h
- /home/jan/Promotion_linuxPC/Optim_VSTRAP/vldn/controller/abstract_validation.cpp

4.3 control_field_class.Arrow3D Class Reference

Inheritance diagram for control_field_class.Arrow3D:



Collaboration diagram for control_field_class.Arrow3D:



Public Member Functions

- def **__init__** (self, xs, ys, zs, args, kwargs)
- def **draw** (self, renderer)

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

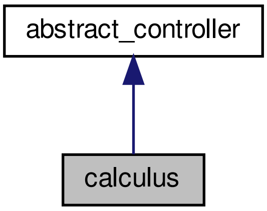
- /home/jan/Promotion_linuxPC/Optim_VSTRAP/optim-vstrap-toolset/toolset/control_field_class.py

4.4 calculus Class Reference

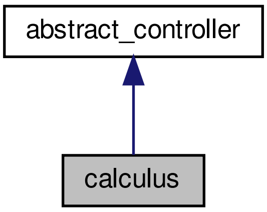
The calculus class provides method from analysis.

```
#include <calculus.h>
```

Inheritance diagram for calculus:



Collaboration diagram for calculus:



Public Member Functions

- double **divergence_vector** (arma::mat input)

Static Public Member Functions

- static std::vector< double > **cross_product** (std::vector< double > v1, std::vector< double > v2)

4.4.1 Detailed Description

The calculus class provides method from analysis.

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

- /home/jan/Promotion_linuxPC/Optim_VSTRAP/src/tools/calculus.h
- /home/jan/Promotion_linuxPC/Optim_VSTRAP/src/tools/calculus.cpp

4.5 mesh.Cell Class Reference

Public Member Functions

- def **__init__** (self)
- def **set_nodes** (self, nodes)
- def **calc_volume** (self, nodes)
- def **calc_barycenter** (self, nodes)

Public Attributes

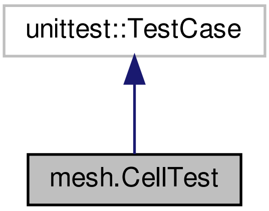
- **id**
- **nodes_ids**
- **value**
- **volume**
- **type**
- **barycenter**

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

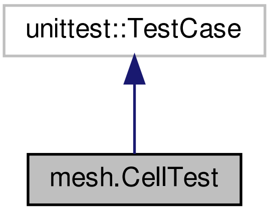
- /home/jan/Promotion_linuxPC/Optim_VSTRAP/optim-vstrap-toolset/toolset/mesh.py

4.6 mesh.CellTest Class Reference

Inheritance diagram for mesh.CellTest:



Collaboration diagram for mesh.CellTest:



Public Member Functions

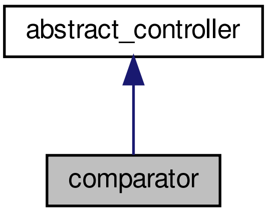
- def **test_calc_volume** (self)

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

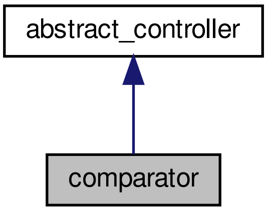
- /home/jan/Promotion_linuxPC/Optim_VSTRAP/optim-vstrap-toolset/tests/mesh.py

4.7 comparator Class Reference

Inheritance diagram for comparator:



Collaboration diagram for comparator:



Public Member Functions

- double **norm_difference_doubleVector** (std::vector< double > v1, std::vector< double > v2)

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

- /home/jan/Promotion_linuxPC/Optim_VSTRAP/src/objects/comparator.h
- /home/jan/Promotion_linuxPC/Optim_VSTRAP/src/objects/comparator.cpp

4.8 control_field_class.Control_field Class Reference

Public Member Functions

- def **__init__** (self)
- def **__str__** (self)
- def **clear** (self)
- def **create_Lists** (self, controlFile, meshFile, scaling)
- def **plot_Control_field** (self, nodesMesh, endPoints, scaling, directorySRC, boxlim)

Public Attributes

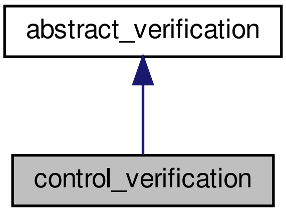
- **control**
- **nodesMesh**
- **endPoints**

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

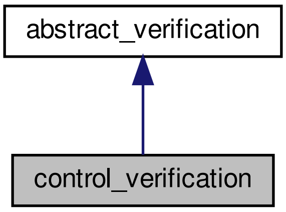
- /home/jan/Promotion_linuxPC/Optim_VSTRAP/optim-vstrap-toolset/toolset/control_field_class.py

4.9 control_verification Class Reference

Inheritance diagram for control_verification:



Collaboration diagram for control_verification:



Static Public Member Functions

- static int **start_verification** (int argc, char ∗∗argv)
- static double **calculate_mean** (arma::mat control)
- static std::vector< double > **calculate_mean_doubleMatrix** (std::vector< std::vector< double >> control)
- static arma::mat **calculate_cross_error** (arma::mat control, arma::mat barycenters, std::vector< double > &valide_vector)

Additional Inherited Members

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

- /home/jan/Promotion_linuxPC/Optim_VSTRAP/vldn/control/control_validation.h
- /home/jan/Promotion_linuxPC/Optim_VSTRAP/vldn/control/control_validation.cpp

4.10 coordinate_phase_space_time Class Reference

The coordinate_phase_space_time class defines coordinates in the seven dimensional time-phase-space cylinder.

```
#include <coordinate_phase_space_time.h>
```

Public Member Functions

- **coordinate_phase_space_time** (int cell_id, int vx, int vy, int vz, int time)
- std::string **toString** () const
- bool **operator==** (const coordinate_phase_space_time &coordinate) const
- coordinate_phase_space_time **operator-** (const coordinate_phase_space_time &coordinate) const
- int **getPx** () const
- void **setPx** (int value)
- int **getPy** () const
- void **setPy** (int value)
- int **getPz** () const
- void **setPz** (int value)
- int **getVx** () const
- void **setVx** (int value)
- int **getVy** () const
- void **setVy** (int value)
- int **getVz** () const
- void **setVz** (int value)
- int **getTime** () const
- void **setTime** (int value)
- int **getCell_id** () const
- void **setCell_id** (int value)

4.10.1 Detailed Description

The coordinate_phase_space_time class defines coordinates in the seven dimensional time-phase-space cylinder.

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

- /home/jan/Promotion_linuxPC/Optim_VSTRAP/src/objects/coordinate_phase_space_time.h
- /home/jan/Promotion_linuxPC/Optim_VSTRAP/src/objects/coordinate_phase_space_time.cpp

4.11 data_provider Class Reference

Public Member Functions

- **data_provider** (const char ∗filename)
- std::map< std::string, std::string > **read_paths** (const char ∗filename)
- std::map< std::string, double > **read_optimization_parameters** (const char ∗filename)
- std::map< std::string, std::string > **read_subroutines** (const char ∗filename)
- std::map< int, std::vector< double > > **read_mesh_barycenters** (const char ∗filename)
- std::map< std::string, std::string > **getPaths** () const
- void **setPaths** (const std::map< std::string, std::string > &value)
- std::map< std::string, double > **getOptimizationParameters** () const
- void **setOptimizationParameters** (const std::map< std::string, double > &value)
- std::map< std::string, std::string > **getSubroutines** () const
- void **setSubroutines** (const std::map< std::string, std::string > &value)
- std::map< int, std::vector< double > > **getMesh_barycenters** () const
- void **setMesh_barycenters** (const std::map< int, std::vector< double > > &value)

Static Public Member Functions

- static arma::mat **convert_barycenters_toArmaMat** (std::map< int, std::vector< double >> barycenters)

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

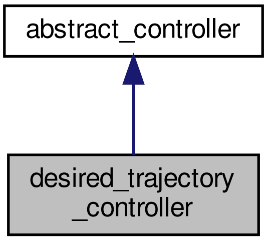
- /home/jan/Promotion_linuxPC/Optim_VSTRAP/src/objects/data_provider.h
- /home/jan/Promotion_linuxPC/Optim_VSTRAP/src/objects/data_provider.cpp

4.12 desired_trajectory_controller Class Reference

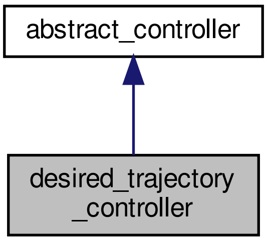
The desired_trajectory_controller class provides the trajectory of the mean value in phase space.

```
#include <desired_trajectory_controller.h>
```

Inheritance diagram for desired_trajectory_controller:



Collaboration diagram for desired_trajectory_controller:



Public Member Functions

- std::vector< double > **trajectory_desired** (std::vector< double > barycenter, unsigned int l, unsigned int m, unsigned int n, unsigned int o, std::vector< std::vector< double > > brockettVector, unsigned int plasma_↩ state_output_interval)
- std::vector< double > trajectory_desired_brockett (std::vector< std::vector< double > > brockettVector, unsigned int o, unsigned int plasma_state_output_interval)

  trajectory_desired_brockett provides the desired trajectory using a time dependent vector as input

4.12.1 Detailed Description

The desired_trajectory_controller class provides the trajectory of the mean value in phase space.

4.12.2 Member Function Documentation

4.12.2.1 trajectory_desired_brockett()

```
std::vector< double > desired_trajectory_controller::trajectory_desired_brockett (
            std::vector< std::vector< double > > brockettVector,
            unsigned int o,
            unsigned int plasma_state_output_interval )
```

trajectory_desired_brockett provides the desired trajectory using a time dependent vector as input

Parameters

brockettVector	
o	
plasma_state_output_interval	

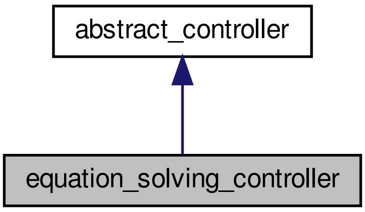
Returns

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

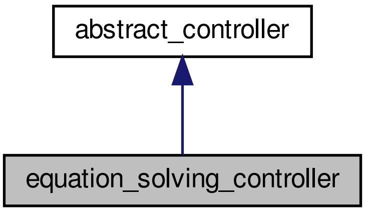
- /home/jan/Promotion_linuxPC/Optim_VSTRAP/src/controller/desired_trajectory_controller.h
- /home/jan/Promotion_linuxPC/Optim_VSTRAP/src/controller/desired_trajectory_controller.cpp

4.13 equation_solving_controller Class Reference

Inheritance diagram for equation_solving_controller:



Collaboration diagram for equation_solving_controller:



Public Member Functions

- int **start_solving_forward** (std::string start_forward)
- int **start_solving_backward** (std::string start_backward)
- arma::mat **D1_second_order** ()
- arma::mat **D1_forward** ()
- arma::mat **D1_backward** ()

- arma::mat **Laplacian_3D** ()
- arma::mat **Laplacian_Squared_3D** ()
- arma::mat **D1X1_second_order** ()
- arma::mat **D1X2_second_order** ()
- arma::mat **D1X3_second_order** ()

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

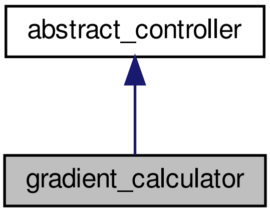
- /home/jan/Promotion_linuxPC/Optim_VSTRAP/src/controller/equation_solving_controller.h
- /home/jan/Promotion_linuxPC/Optim_VSTRAP/src/controller/equation_solving_controller.cpp

4.14 gradient_calculator Class Reference

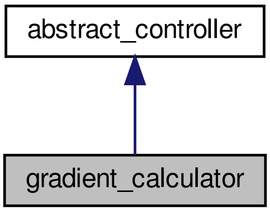
The gradient_calculator class provides method for assembling to gradient, which is used in the calculation of the new step-direction for controls in H^2 Sobolev-space.

```
#include <gradient_calculator.h>
```

Inheritance diagram for gradient_calculator:



Collaboration diagram for gradient_calculator:



Public Member Functions

- **gradient_calculator** (const char ∗filename)
- arma::mat calculateGradient_forceControl_space_Hm_not_parallel (std::vector< std::unordered_map< coordinate_phase_space_time, double >> forwardPDF_time, std::vector< std::unordered_map< coordinate_phase_space_time, double >> backwardPDF_time, arma::mat control)

  calculateGradient_forceControl_space_Hm_not_parallel calculates the gradient without using any parallelization;
- arma::mat calculateGradient_forceControl_space_Hm (std::vector< std::unordered_map< coordinate_←phase_space_time, double >> forwardPDF_time, std::vector< std::unordered_map< coordinate_phase←_space_time, double >> backwardPDF_time, arma::mat control)

  calculateGradient_forceControl_space_Hm calculates the gradient with parallelization
- arma::mat calculateGradient_forceControl_space_Hm_plasma (std::vector< std::unordered_map< coordinate_phase_space_time, double >> forwardPDF_time, std::vector< std::unordered_map< coordinate_phase_space_time, double >> backwardPDF_time, std::vector< std::unordered_map< coordinate_phase_space_time, double >> forwardPDF_time_electrons, std::vector< std::unordered_←map< coordinate_phase_space_time, double >> backwardPDF_time_electrons, arma::mat control)

  calculateGradient_forceControl_space_Hm_plasma calculates the gradient with two different species (ions, electrons) present

4.14.1 Detailed Description

The gradient_calculator class provides method for assembling to gradient, which is used in the calculation of the new step-direction for controls in H^2 Sobolev-space.

4.14.2 Member Function Documentation

4.14.2.1 calculateGradient_forceControl_space_Hm()

```
arma::mat gradient_calculator::calculateGradient_forceControl_space_Hm (
            std::vector< std::unordered_map< coordinate_phase_space_time, double >> forward←
PDF_time,
            std::vector< std::unordered_map< coordinate_phase_space_time, double >> backward←
PDF_time,
            arma::mat control )
```

calculateGradient_forceControl_space_Hm calculates the gradient with parallelization

Parameters

forwardPDF_time	
backwardPDF_time	
control	

Returns

4.14.2.2 calculateGradient_forceControl_space_Hm_not_parallel()

```
arma::mat gradient_calculator::calculateGradient_forceControl_space_Hm_not_parallel (
            std::vector< std::unordered_map< coordinate_phase_space_time, double >> forward↩
PDF_time,
            std::vector< std::unordered_map< coordinate_phase_space_time, double >> backward↩
PDF_time,
            arma::mat control )
```

calculateGradient_forceControl_space_Hm_not_parallel calculates the gradient without using any parallelization;

Parameters

forwardPDF_time	
backwardPDF_time	
control	

Returns

4.14.2.3 calculateGradient_forceControl_space_Hm_plasma()

```
arma::mat gradient_calculator::calculateGradient_forceControl_space_Hm_plasma (
            std::vector< std::unordered_map< coordinate_phase_space_time, double >> forward↩
PDF_time,
            std::vector< std::unordered_map< coordinate_phase_space_time, double >> backward↩
PDF_time,
            std::vector< std::unordered_map< coordinate_phase_space_time, double >> forward↩
PDF_time_electrons,
            std::vector< std::unordered_map< coordinate_phase_space_time, double >> backward↩
PDF_time_electrons,
            arma::mat control )
```

calculateGradient_forceControl_space_Hm_plasma calculates the gradient with two different species (ions, electrons) present

Parameters

forwardPDF_time	
backwardPDF_time	
forwardPDF_time_electrons	
backwardPDF_time_electrons	
control	

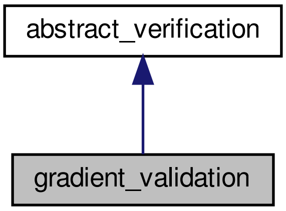
Returns

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

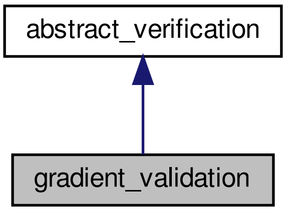
- /home/jan/Promotion_linuxPC/Optim_VSTRAP/src/optimization/gradient_calculator.h
- /home/jan/Promotion_linuxPC/Optim_VSTRAP/src/optimization/gradient_calculator.cpp

4.15 gradient_validation Class Reference

Inheritance diagram for gradient_validation:



Collaboration diagram for gradient_validation:



Static Public Member Functions

- static int **landau_validation** (int argc, char ∗∗argv)

Additional Inherited Members

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

- /home/jan/Promotion_linuxPC/Optim_VSTRAP/vldn/gradient/gradient_validation.h
- /home/jan/Promotion_linuxPC/Optim_VSTRAP/vldn/gradient/gradient_validation.cpp

4.16 std::hash< coordinate_phase_space_time > Struct Template Reference

Public Types

- typedef coordinate_phase_space_time **argument_type**
- typedef size_t **result_type**

Public Member Functions

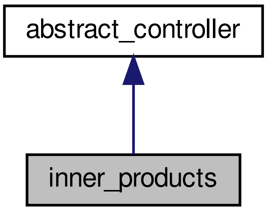
- size_t **operator()** (const argument_type &x) const

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

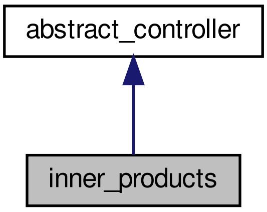
- /home/jan/Promotion_linuxPC/Optim_VSTRAP/src/objects/coordinate_phase_space_time.h

4.17 inner_products Class Reference

Inheritance diagram for inner_products:



Collaboration diagram for inner_products:



Public Member Functions

- double **L2_inner_product** (arma::mat m1, arma::mat m2)
- double H1_inner_product (arma::mat m1, arma::mat m2)
- double H2_inner_product (arma::mat m1, arma::mat m2)

4.17.1 Member Function Documentation

4.17.1.1 H1_inner_product()

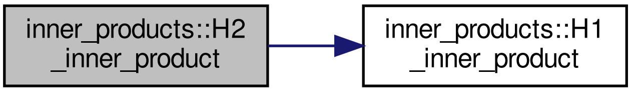
```
double inner_products::H1_inner_product (
            arma::mat m1,
            arma::mat m2 )
```

L2 part

4.17.1.2 H2_inner_product()

```
double inner_products::H2_inner_product (
            arma::mat m1,
            arma::mat m2 )
```

L2 and H1 partHere is the call graph for this function:

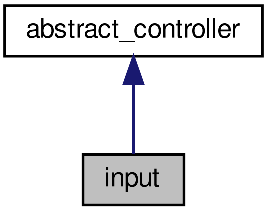


The documentation for this class was generated from the following files:

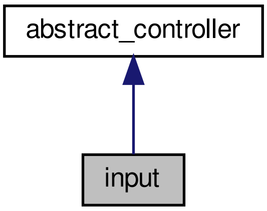
- /home/jan/Promotion_linuxPC/Optim_VSTRAP/src/tools/inner_products.h
- /home/jan/Promotion_linuxPC/Optim_VSTRAP/src/tools/inner_products.cpp

4.18 input Class Reference

Inheritance diagram for input:



Collaboration diagram for input:



Public Member Functions

- unsigned int **read_plasma_state_forward** (std::vector< std::vector< particle >> &forwardParticles, std::string file_name)
- unsigned int **read_plasma_state_backward** (std::vector< std::vector< particle >> &backwardParticles, std::string file_name)
- arma::mat readControl (const char ∗filename, int pcell_gp)

  readControl reads in control cells (control in volume, xml format)

Static Public Member Functions

- static std::vector< particle > **readParticleVector** (std::string filename, std::string delimiter)
- static std::vector< std::vector< double > > **readDoubleMatrix** (std::string filename, int pcell_gp, std::string delimiter)
- static std::vector< double > **readDoubleVector** (const char ∗filename)
- static std::vector< std::vector< double > > readBrockettFile (std::string filename, std::string delimiter, unsigned int lines)

  readBrockettFile reads file with time-dependent desired trajectory of the mean

4.18.1 Member Function Documentation

4.18.1.1 readBrockettFile()

```
std::vector< std::vector< double > > input::readBrockettFile (
            std::string filename,
            std::string delimiter,
            unsigned int lines )  [static]
```

readBrockettFile reads file with time-dependent desired trajectory of the mean

Parameters

filename	
delimiter	
lines	

Returns

4.18.1.2 readControl()

```
arma::mat input::readControl (
            const char * filename,
            int pcell_gp )
```

readControl reads in control cells (control in volume, xml format)

Parameters

filename	
pcell_gp	

Returns

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

- /home/jan/Promotion_linuxPC/Optim_VSTRAP/src/io/input.h
- /home/jan/Promotion_linuxPC/Optim_VSTRAP/src/io/input.cpp

4.19 mesh.Mesh Class Reference

Public Member Functions

- def **__init__** (self)
- def **__str__** (self)
- def **clear** (self)
- def **read_mesh_xml** (self, file_name)
- def **interpolate_cell2node** (self)
- def **read_control_csv** (self, file_name)
- def **read_control_xml** (self, file_name)
- def **write_control_csv** (self, file_name)
- def **write_control_xml** (self, file_name, control_type)
- def **write_barycenters_xml** (self, file_name)

Public Attributes

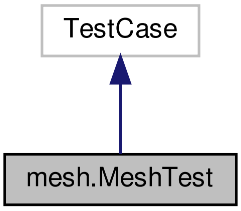
- **cells**
- **nodes**
- **volume**

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

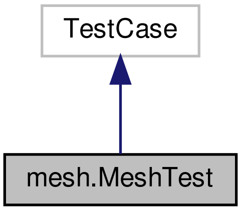
- /home/jan/Promotion_linuxPC/Optim_VSTRAP/optim-vstrap-toolset/toolset/mesh.py

4.20 mesh.MeshTest Class Reference

Inheritance diagram for mesh.MeshTest:



Collaboration diagram for mesh.MeshTest:



Public Member Functions

- def **test_read_mesh_xml** (self)
- def **test_read_control_csv** (self)
- def **test_read_control_xml** (self)
- def **test_interpolate_cell2node** (self)

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

- /home/jan/Promotion_linuxPC/Optim_VSTRAP/optim-vstrap-toolset/tests/mesh.py

4.21 mesh.Node Class Reference

Public Member Functions

- def **__init__** (self, id=0, coord=(0.0, 0.0, 0.0))
- def **get_position** (self)

Public Attributes

- **id**
- **x_coord**
- **y_coord**
- **z_coord**
- **value**

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

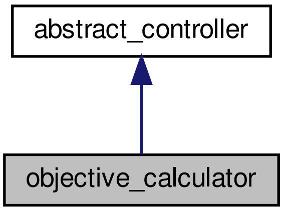
- /home/jan/Promotion_linuxPC/Optim_VSTRAP/optim-vstrap-toolset/toolset/mesh.py

4.22 objective_calculator Class Reference

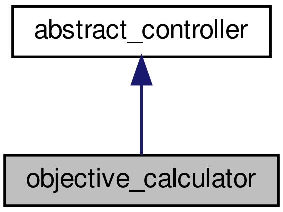
The objective_calculator class calculates the objective/functional according to Brockett's approach of ensemble optimal control problems; see, e.g., Bartsch, J., Borzì, A., Fanelli, F. et al. A theoretical investigation of Brockett's ensemble optimal control problems. Calc. Var. 58, 162 (2019). https://doi.org/10.↵1007/s00526-019-1604-2.

```
#include <objective_calculator.h>
```

Inheritance diagram for objective_calculator:



Collaboration diagram for objective_calculator:



Public Member Functions

- **objective_calculator** (const char ∗filename)
- double **calculate_objective** (std::vector< std::unordered_map< coordinate_phase_space_time, double >> forwardPDF_time, arma::mat control)

4.22.1 Detailed Description

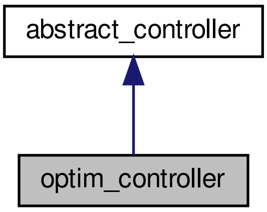
The objective_calculator class calculates the objective/functional according to Brockett's approach of ensemble optimal control problems; see, e.g., Bartsch, J., Borzì, A., Fanelli, F. et al. A theoretical investigation of Brockett's ensemble optimal control problems. Calc. Var. 58, 162 (2019). https://doi.org/10.↩1007/s00526-019-1604-2.

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

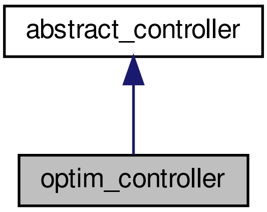
- /home/jan/Promotion_linuxPC/Optim_VSTRAP/src/optimization/objective_calculator.h
- /home/jan/Promotion_linuxPC/Optim_VSTRAP/src/optimization/objective_calculator.cpp

4.23 optim_controller Class Reference

Inheritance diagram for optim_controller:



Collaboration diagram for optim_controller:



Public Member Functions

- int start_optimizer (int argc, const char $**$argv)
  start_optimizer reads in the command line command and starts the optimizer

Static Public Member Functions

- static int main_optimization_algorithm (const char ∗input_xml_path)

  main_optimization_algorithm is the core optimization algorithm which uses the paramteres defined in the input file for the optimizer

4.23.1 Member Function Documentation

4.23.1.1 main_optimization_algorithm()

```
int optim_controller::main_optimization_algorithm (
             const char ∗ input_xml_path )  [static]
```

main_optimization_algorithm is the core optimization algorithm which uses the paramteres defined in the input file for the optimizer

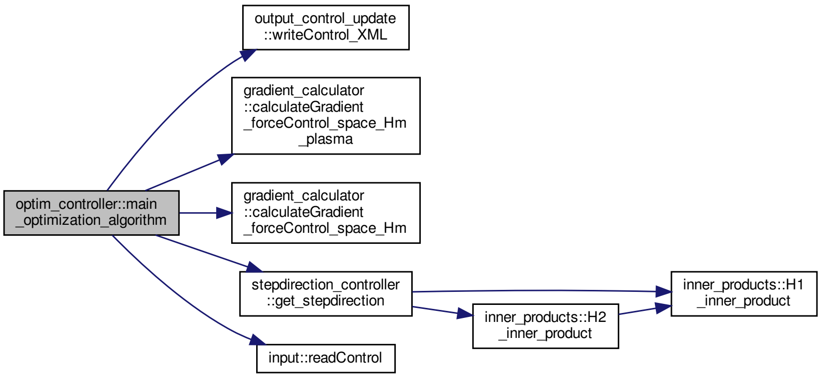
Parameters

input_xml_path	

Returns

START OPTIMIZATION ITERATION

first stepsize guess, scaled with norm of gradientHere is the call graph for this function:


output_control_update ::writeControl_XML
gradient_calculator ::calculateGradient _forceControl_space_Hm _plasma
optim_controller::main _optimization_algorithm
gradient_calculator ::calculateGradient _forceControl_space_Hm
stepdirection_controller ::get_stepdirection
inner_products::H1 _inner_product
inner_products::H2 _inner_product
input::readControl

4.23.1.2 start_optimizer()

```
int optim_controller::start_optimizer (
             int argc,
             const char ** argv )
```

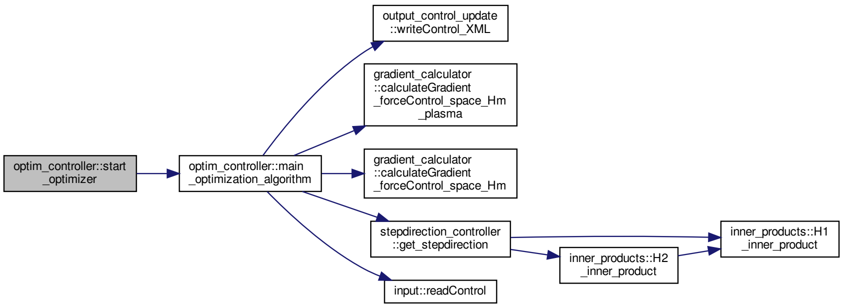
start_optimizer reads in the command line command and starts the optimizer

Parameters

argc	
argv	

Returns

Here is the call graph for this function:


output_control_update ::writeControl_XML
gradient_calculator ::calculateGradient _forceControl_space_Hm _plasma
optim_controller::start _optimizer
optim_controller::main _optimization_algorithm
gradient_calculator ::calculateGradient _forceControl_space_Hm
stepdirection_controller ::get_stepdirection
inner_products::H1 _inner_product
inner_products::H2 _inner_product
input::readControl

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

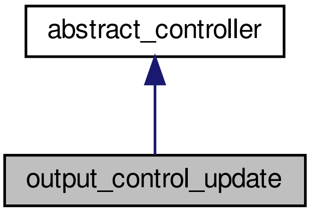
- /home/jan/Promotion_linuxPC/Optim_VSTRAP/src/controller/optim_controller.h
- /home/jan/Promotion_linuxPC/Optim_VSTRAP/src/controller/optim_controller.cpp

4.24 output_control_update Class Reference

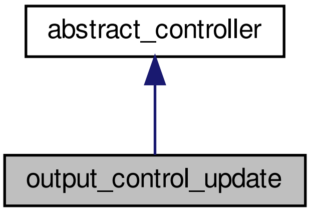
The output_control_update class offers functions to write the update of the control in a file that is readable by the solver for forward and backward equation.

```
#include <output_control_update.h>
```

Inheritance diagram for output_control_update:



Collaboration diagram for output_control_update:



Public Member Functions

- **output_control_update** (const char ∗filename)
- int writeControl_XML (arma::mat control)

  writeControl_XML takes a control and writes a corresponding XML file
- int **writeArmaMatrixToFile** (arma::mat input, std::string filename)

Static Public Member Functions

- static int **interpolate_control** (data_provider provider)

4.24.1 Detailed Description

The output_control_update class offers functions to write the update of the control in a file that is readable by the solver for forward and backward equation.

4.24.2 Member Function Documentation

4.24.2.1 writeControl_XML()

```
int output_control_update::writeControl_XML (
            arma::mat control )
```

writeControl_XML takes a control and writes a corresponding XML file

Parameters

control	(arma::mat)

Returns

0 if processed successfully

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

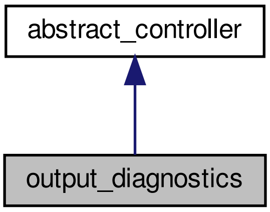
- /home/jan/Promotion_linuxPC/Optim_VSTRAP/src/io/output_control_update.h
- /home/jan/Promotion_linuxPC/Optim_VSTRAP/src/io/output_control_update.cpp

4.25 output_diagnostics Class Reference

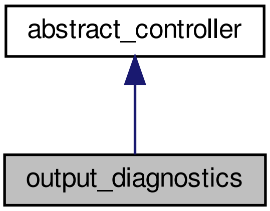
The output_diagnostics class writes the value of different objects to txt files.

```
#include <output_diagnostics.h>
```

Inheritance diagram for output_diagnostics:



Collaboration diagram for output_diagnostics:



Public Member Functions

- int **writeArmaMatrixToFile** (arma::mat gradient, std::string filename)
- int **writeDoubleToFile** (double value, std::string filename)
- int **writeDoubleVectorToFile** (std::vector< double > vector, std::string filename)

4.25.1 Detailed Description

The output_diagnostics class writes the value of different objects to txt files.

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

- /home/jan/Promotion_linuxPC/Optim_VSTRAP/src/io/output_diagnostics.h
- /home/jan/Promotion_linuxPC/Optim_VSTRAP/src/io/output_diagnostics.cpp

4.26 parameter_sanity Class Reference

The parameter_sanity class provides sanity checks for parameters definied in the input file of the optimizer.

```
#include <parameter_sanity.h>
```

Public Member Functions

- int **check_adjoint_velocity** (data_provider provider)
- int **check_velocity_discretization** (data_provider provider)

4.26.1 Detailed Description

The parameter_sanity class provides sanity checks for parameters definied in the input file of the optimizer.

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

- /home/jan/Promotion_linuxPC/Optim_VSTRAP/src/tools/parameter_sanity.h
- /home/jan/Promotion_linuxPC/Optim_VSTRAP/src/tools/parameter_sanity.cpp

4.27 particle Class Reference

Public Member Functions

- **particle** (double vx, double vy, double vz)
- **particle** (double px, double py, double pz, double vx, double vy, double vz)
- **particle** (double px, double py, double pz, double vx, double vy, double vz, int cell_id)
- bool **operator==** (const particle &particle) const
- double getVelocityMagnitudeParticle ()

  getVelocityMagnitudeParticle calculates speed of particles using Euclidean Norm
- std::string toString ()

  toString
- double **getPx** () const
- void **setPx** (double value)
- double **getPy** () const
- void **setPy** (double value)
- double **getPz** () const
- void **setPz** (double value)
- double **getVx** () const
- void **setVx** (double value)
- double **getVy** () const
- void **setVy** (double value)
- double **getVz** () const
- void **setVz** (double value)
- int **getCell_id** () const
- void **setCell_id** (int value)
- double **getWeight** () const
- void **setWeight** (double value)

4.27.1 Member Function Documentation

4.27.1.1 getVelocityMagnitudeParticle()

```
double particle::getVelocityMagnitudeParticle ( )
```

getVelocityMagnitudeParticle calculates speed of particles using Euclidean Norm

Returns

4.27.1.2 toString()

```
std::string particle::toString ( )
```

toString

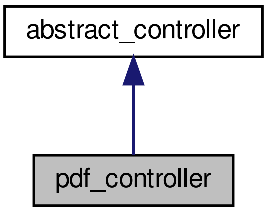
Returns

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

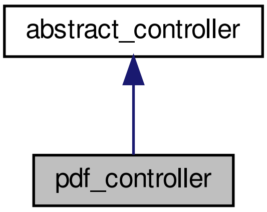
- /home/jan/Promotion_linuxPC/Optim_VSTRAP/src/objects/particle.h
- /home/jan/Promotion_linuxPC/Optim_VSTRAP/src/objects/particle.cpp

4.28 pdf_controller Class Reference

Inheritance diagram for pdf_controller:



Collaboration diagram for pdf_controller:



Public Member Functions

- int **assemblingMultiDim** (std::vector< std::vector< particle > > &particlesTime, unsigned int equationType, std::vector< std::unordered_map< coordinate_phase_space_time, double > > &pdf_time)
- int **assemblingMultiDim_parallel** (std::vector< std::vector< particle > > &particlesTime, unsigned int equationType, std::vector< std::unordered_map< coordinate_phase_space_time, double > > &pdf_time)
- std::vector< std::vector< std::vector< std::vector< double > > > > **relaxating_GaussSeidel_4D** (std::vector< std::vector< std::vector< std::vector< double >>>> pdf, unsigned int numberOfRelaxationSteps)
- double **calculate_wasserstein_metric** (std::vector< std::vector< particle >> dist1, std::vector< std::vector< particle >> dist2)
- double **calculate_wasserstein_metric_histogramm** (std::vector< std::unordered_map< coordinate_phase_space_time, double >> dist1, std::vector< std::unordered_map< coordinate_phase_space_time, double >> dist2)

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

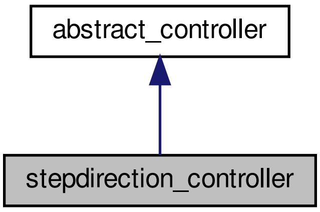
- /home/jan/Promotion_linuxPC/Optim_VSTRAP/src/controller/pdf_controller.h
- /home/jan/Promotion_linuxPC/Optim_VSTRAP/src/controller/pdf_controller.cpp

4.29 stepdirection_controller Class Reference

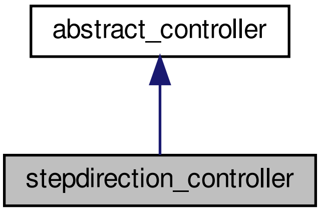
The stepdirection_controller class provides different methods for finding the step-direction, as gradient descent and NCG schemes with different update rules.

```
#include <stepdirection_controller.h>
```

Inheritance diagram for stepdirection_controller:



Collaboration diagram for stepdirection_controller:



Public Member Functions

- **stepdirection_controller** (const char ∗filename)
- arma::mat get_stepdirection (arma::mat gradient, arma::mat gradient_old, arma::mat stepdirectionOld, unsigned int optimization_iteration)

  get_stepdirection generic method called in the main optimizer algorithm

4.29.1 Detailed Description

The stepdirection_controller class provides different methods for finding the step-direction, as gradient descent and NCG schemes with different update rules.

4.29.2 Member Function Documentation

4.29.2.1 get_stepdirection()

```
arma::mat stepdirection_controller::get_stepdirection (
            arma::mat gradient,
            arma::mat gradient_old,
            arma::mat stepdirectionOld,
            unsigned int optimization_iteration )
```

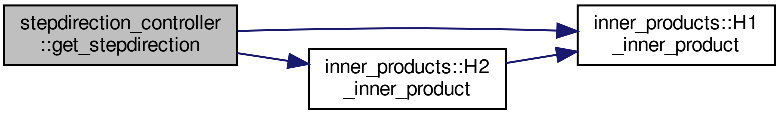
get_stepdirection generic method called in the main optimizer algorithm

Parameters

gradient	
gradient_old	
stepdirectionOld	
optimization_iteration	

Returns

Here is the call graph for this function:



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

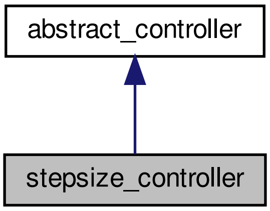
- /home/jan/Promotion_linuxPC/Optim_VSTRAP/src/optimization/stepdirection_controller.h
- /home/jan/Promotion_linuxPC/Optim_VSTRAP/src/optimization/stepdirection_controller.cpp

4.30 stepsize_controller Class Reference

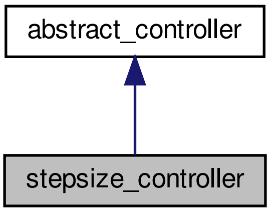
The stepsize_controller class provides different methods for finding an accepted step-size (resulting in a decreasing value of the functional)

```
#include <stepsize_controller.h>
```

Inheritance diagram for stepsize_controller:



Collaboration diagram for stepsize_controller:



Public Member Functions

- **stepsize_controller** (const char ∗filename)
- int **calculate_stepsize** (arma::mat &gradient, double J0, arma::mat &control, arma::mat &stepdirection, std::vector< particle > &inputParticles, double &stepsize0)

4.30.1 Detailed Description

The stepsize_controller class provides different methods for finding an accepted step-size (resulting in a decreasing value of the functional)

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

- /home/jan/Promotion_linuxPC/Optim_VSTRAP/src/optimization/stepsize_controller.h
- /home/jan/Promotion_linuxPC/Optim_VSTRAP/src/optimization/stepsize_controller.cpp