CloudTracker

Generated by Doxygen 1.11.0

1 CloudTracker 1

1.1 Table of Contents . . . 1
1.2 Installation . . . 1
1.3 Usage . . . 2
1.4 Project Structure . . . 2
1.5 Modules . . . 2
1.5.1 Matcher . . . 2
1.5.2 Linker . . . 2

2 Class Index 3

2.1 Class List . . . 3

3 File Index 5

3.1 File List . . . 5

4 Class Documentation 7

4.1 ChildParentInstance Class Reference . . . 7
4.1.1 Constructor & Destructor Documentation . . . 7
4.1.1.1 ChildParentInstance() . . . 7
4.2 CitySnaps Class Reference . . . 8
4.2.1 Constructor & Destructor Documentation . . . 8
4.2.1.1 CitySnaps() . . . 8
4.3 CloudGroup Struct Reference . . . 8
4.3.1 Member Function Documentation . . . 9
4.3.1.1 add_member() . . . 9
4.4 CloudStruct Struct Reference . . . 9
4.5 Group_struct Struct Reference . . . 10
4.6 MemberCloud Class Reference . . . 10
4.6.1 Constructor & Destructor Documentation . . . 10
4.6.1.1 MemberCloud() . . . 10
4.6.2 Member Function Documentation . . . 11
4.6.2.1 add_child() . . . 11
4.6.2.2 add_parent() . . . 11
4.7 Params Struct Reference . . . 12

5 File Documentation 13

5.1 io.h . . . 13
5.2 io.h . . . 13
5.3 linker_functions.h . . . 14
5.4 read_params.h . . . 14
5.5 read_params.h . . . 14
5.6 structs_and_classes.h . . . 15
5.7 include/matcher/structs_and_classes.h File Reference . . . 15
5.7.1 Detailed Description . . . 16

5.7.2 Function Documentation . . . 16
5.7.2.1 create_group() . . . 16
5.8 structs_and_classes.h . . . 17
5.9 utilities.h . . . 18
5.10 utilities.h . . . 18
5.11 matcher_functions.h . . . 18
5.12 src/linker/io/io_hdf5.cpp File Reference . . . 19
5.12.1 Detailed Description . . . 19
5.12.2 Function Documentation . . . 19
5.12.2.1 child_info() . . . 19
5.12.2.2 get_child_list() . . . 20
5.12.2.3 get_last_group() . . . 20
5.12.2.4 read_cloud_data_double() . . . 21
5.12.2.5 read_mass() . . . 21
5.13 src/linker/io/io_txt.cpp File Reference . . . 21
5.13.1 Detailed Description . . . 22
5.13.2 Function Documentation . . . 22
5.13.2.1 print_list() . . . 22
5.13.2.2 write_to_file() . . . 22
5.14 src/linker/io/read_params.cpp File Reference . . . 23
5.14.1 Detailed Description . . . 23
5.14.2 Function Documentation . . . 23
5.14.2.1 parseParams() . . . 23
5.14.2.2 printParams() . . . 24
5.14.2.3 trim() . . . 24
5.15 src/matcher/io/read_params.cpp File Reference . . . 24
5.15.1 Detailed Description . . . 25
5.15.2 Function Documentation . . . 25
5.15.2.1 parseParams() . . . 25
5.15.2.2 printParams() . . . 25
5.15.2.3 trim() . . . 26
5.16 src/linker/main/linker.cpp File Reference . . . 26
5.16.1 Detailed Description . . . 26
5.16.2 Function Documentation . . . 27
5.16.2.1 main() . . . 27
5.17 src/linker/main/linker_functions.cpp File Reference . . . 27
5.17.1 Detailed Description . . . 28
5.17.2 Function Documentation . . . 28
5.17.2.1 check_if_cloud_exists_in_list() . . . 28
5.17.2.2 find_proper_child() . . . 28
5.17.2.3 linker() . . . 29
5.17.2.4 load_to_cloudlist() . . . 29

5.17.2.5 load_to_tracked_cloud_list() . . . 29
5.18 src/linker/utils/structs_and_classes.cpp File Reference . . . 30
5.18.1 Detailed Description . . . 30
5.18.2 Function Documentation . . . 30
5.18.2.1 create_group() . . . 30
5.18.2.2 initialize_cloud() . . . 31
5.19 src/matcher/utils/structs_and_classes.cpp File Reference . . . 31
5.19.1 Detailed Description . . . 32
5.19.2 Function Documentation . . . 32
5.19.2.1 create_group() . . . 32
5.20 src/linker/utils/utilities.cpp File Reference . . . 32
5.20.1 Detailed Description . . . 33
5.20.2 Function Documentation . . . 33
5.20.2.1 check_if_exists() . . . 33
5.20.2.2 find_num_clouds() . . . 33
5.20.2.3 get_cloud_name() . . . 34
5.20.2.4 get_num_particles() . . . 34
5.20.2.5 get_snapshot_name() . . . 34
5.21 src/matcher/utils/utilities.cpp File Reference . . . 35
5.21.1 Detailed Description . . . 35
5.21.2 Function Documentation . . . 36
5.21.2.1 check_if_value_exists_in_array() . . . 36
5.21.2.2 find_num_clouds() . . . 36
5.21.2.3 get_first_index() . . . 36
5.21.2.4 get_snapshot_name() . . . 36
5.21.2.5 print_array_double() . . . 37
5.21.2.6 print_array_int() . . . 37
5.21.2.7 sum_array() . . . 37
5.22 src/matcher/io/io.cpp File Reference . . . 38
5.22.1 Detailed Description . . . 38
5.22.2 Function Documentation . . . 39
5.22.2.1 get_last_group() . . . 39
5.22.2.2 read_cloud_data_double() . . . 39
5.22.2.3 read_cloud_data_int() . . . 39
5.22.2.4 write_to_hdf5_file() . . . 40
5.22.2.5 writeCloudData() . . . 40
5.22.2.6 writeGroupDataChildren() . . . 40
5.22.2.7 writeGroupDataParents() . . . 41
5.23 src/matcher/main/matcher.cpp File Reference . . . 41
5.23.1 Detailed Description . . . 42
5.23.2 Function Documentation . . . 42
5.23.2.1 main() . . . 42

5.24 src/matcher/main/matcher_functions.cpp File Reference 42
5.24.1 Detailed Description 43
5.24.2 Function Documentation 43
5.24.2.1 compare_particles() 43
5.24.2.2 get_cloud_name() 44
5.24.2.3 load_clouds_to_group() 44
5.24.2.4 mass_frac() 44
5.24.2.5 matcher() 45

Chapter 1

CloudTracker

CloudTracker is a tool designed for tracking collections of particles/cells (called clouds for the purposes of this code) in fluid simulations. Originally designed for galaxy formation simulations, but can be applied to other contexts easily. It's a tool that can be used in post processing to track collections that have already been identified. You would have to first use some other tool to identify collections of particles/cells in each snapshot, save them in an HDF5 file with a specific format (see Documentation for details.)

It works in two steps: 1) It matches (see Matcher) clouds in two snapshots and identifies parent-child edges (if you think of it as a graph). It currently does so for consecutive snapshots, but the code can be modified easily to have arbitrary spacing between snapshots. 2) It links (see Linker) clouds identified as a parent-child pair (a parent can have more than one child, see below for details on the algorithm). The code follows the algorithm listed below, but would have to be modified if you want a different method. One can write python code to work with matcher outputs and that may be easier if you want a different way of linking clouds together in chains.

The code has been divided into these two parts so that it is easier to maintain more control over each part of the process. It is useful if someone wants to use just a part of the code (like the matcher), but wants to define their own way of linking entities together.

1.1 Table of Contents

- Installation
- Usage
- Project Structure
- Modules
  - Linker
  - Matcher
- Documentation

1.2 Installation

To install CloudTracker, just clone the repository:

```
git clone https://github.com/yourusername/CloudTracker.git
cd CloudTracker
```

You will have to configure the Makefile and add the paths to the required libraries. Use Makefile.systype and Makefile to store settings for your machine. If you're running this on a Mac, the default paths provided should work if you install hdf5 by doing

```
brew install hdf5
```

1.3 Usage

Once you have everything configured, you
```
make
```

1.4 Project Structure

```
|CloudTracker/
|---|include/             # Contains header files
|---|docs/                # Documentation files
|---|src/                 # Source code files
|   |---|linker/          # Linker related source code
|   |   |---|io/          # I/O handling code for linker
|   |   |---|utils/       # Utility functions for linker
|   |   |---|main/        # Main linker functionality
|   |   |---|Makefile     # Makefile for building the linker program
|   |---|matcher/         # Matcher related source code
|   |   |---|io/          # I/O handling code for matcher
|   |   |---|utils/       # Utility functions for matcher
|   |   |---|main/        # Main matcher functionality
|   |   |---|Makefile     # Makefile for building the matcher program
|   |---|Makefile.systype # Makefile for system type detection
|---|README.md            # Project README file
```

1.5 Modules

1.5.1 Matcher

This part of the code will match clouds in subsequent snapshots by checking for the particles of each pair of clouds. This is an $\mathcal{O}(n^2)$ calculation where n denotes the number of clouds in a snapshot. Fortunately, it is still very fast (thanks to being written in C++). This process is easily parallelizable, but currently serial. Parallel support with OpenMP will be added in the future. The matcher program will create HDF5 files which contain information about the children and the amount of "mass" they derive from each of their parents (a parent is any cloud that donates at least one particle to the child cloud). It also contains information about the parents and the amount of "mass" they give to each of their child cloud.

1.5.2 Linker

This part of the code will link together the clouds that matcher matches. The exact algorithm is described in a companion paper, but briefly it proceeds as follows:

- Start from the first snapshot, snapshot A and load all clouds to THE LIST (think of it as a list of lists) in descending order by mass.
- Proceed down the list, for each cloud X identify a "proper child" in snapshot B and add it to the list of descendants of cloud X, which is a part of THE LIST.
  - If cloud X has more than one child, we choose a child Y with the most mass donated from the parent.
  - If child cloud Y is a child of a more massive parent and already exists in THE LIST, the next most massive child of cloud X is chosen.
- When a "proper child" of a cloud is not found, the bloodline ends and no further descendants of that cloud exist.
- If there are clouds in snapshot B that are not a descendant of any cloud X in snapshot A, then add them to THE LIST.

In this method of linking clouds, if two clouds undergo a merger, we consider the less massive cloud to be dead.

Chapter 2

Class Index

2.1 Class List

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

ChildParentInstance 7
CitySnaps 8
CloudGroup 8
CloudStruct 9
Group_struct 10
MemberCloud 10
Params 12

Chapter 3

File Index

3.1 File List

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

include/linker/io.h . . . 13
include/linker/linker_functions.h . . . 14
include/linker/read_params.h . . . 14
include/linker/structs_and_classes.h . . . 15
include/linker/utilities.h . . . 18
include/matcher/io.h . . . 13
include/matcher/matcher_functions.h . . . 18
include/matcher/read_params.h . . . 14
include/matcher/structs_and_classes.h
Contains the definitions of the structs and classes used in the CloudTracker program . . . 15
include/matcher/utilities.h . . . 18
src/linker/io/io_hdf5.cpp
Contains functions to read data from HDF5 files . . . 19
src/linker/io/io_txt.cpp
Contains functions to read and write data to text files. All the tracked cloud list info is stored in text files. Linker generates two text files, one for the names of the clouds and the other for the masses of the clouds . . . 21
src/linker/io/read_params.cpp
Contains functions to read parameters from a file The parameters are stored in a text file and read into a struct Linker then reads these parameters to perform the linking process . . . 23
src/linker/main/linker.cpp
Main driver for the linker part of the code which links clouds in different snapshots together . . . 26
src/linker/main/linker_functions.cpp
Contains the functions used in linking the clouds together . . . 27
src/linker/utils/structs_and_classes.cpp
Contains the structs and classes used in the linker program . . . 30
src/linker/utils/utilities.cpp
Contains some generic utilities like finding the number of clouds in a snapshot, getting the name of the snapshot or getting the name of the cloud if you have a number. Also contains functions for checking two arrays and seeing whether they have something in common, and getting the number of particles in a cloud . . . 32

src/matcher/io/io.cpp
This file contains the definitions of the functions used to write and read data to and from HDF5 files. The write_to_hdf5_file function writes the data to an HDF5 file. The writeCloudData function writes the cloud data to the HDF5 file. The writeGroupDataChildren function writes the children data to the HDF5 file. The writeGroupDataParents function writes the parents data to the HDF5 file. The get_last_group function gets the last group in the group structure. The read_cloud_↩ data_int function reads integer data from the HDF5 file. The read_cloud_data_double function reads double data from the HDF5 file 38
src/matcher/io/read_params.cpp
This file contains the implementation of the functions that read in the parameters from the parameter file. The parameters are stored in a struct called Params. The functions are used in the main driver to read in the parameters and assign them to the code variables 24
src/matcher/main/matcher.cpp
This is the main driver of the matcher application. Its purpose is to match the particles among different cloud entities. It reads in the parameters from the xyz_params.txt file and assigns . . 41
src/matcher/main/matcher_functions.cpp
Contains the definitions of the functions used in the matcher application. This includes the compare_particles, mass_frac, get_cloud_name, and load_clouds_to_group functions. The compare_particles function compares two particle ID lists and counts the number of matching IDs. The mass_frac function calculates the mass fraction of particles between two clouds. The get_cloud_name function returns the name of the cloud based on the cloud number. The load_clouds_to_group function loads the cloud data into the parent or child group. The matcher function is the main driver of the matcher application. It matches the particles among different cloud entities 42
src/matcher/utils/structs_and_classes.cpp
This file contains the definitions of the classes and structures used in the matcher application. This includes the Group_struct, Params, ChildParentInstance, MemberCloud, and CitySnaps classes. The Group_struct class is used to read in the group structure from the HDF5 file. The ChildParentInstance class is used to store information about a child cloud and its parent.
The MemberCloud class is used to store information about a cloud entity in a snapshot. The CitySnaps class is used to store information about clouds in a city (combination of two snapshots). The CloudGroup class is a group of clouds in a city 31
src/matcher/utils/utilities.cpp
This file contains utility functions for the matcher application. This includes functions to check if a double exists in a vector, get the first index of a double in a vector, print a double array, print an int array, sum the elements of a double array, get the name of the snapshot part of a Cloud, find the number of clouds in a snapshot, and get the name of a snapshot as a string with leading zeros to make it a 3-digit number. If you are working with data that is not the default structure, you will have to alter the get_snapshot_name() and find_num_clouds() functions 35

Chapter 4

Class Documentation

4.1 ChildParentInstance Class Reference

Public Member Functions

- ChildParentInstance (std::string name1, double parents_mass_frac_to_child1, double childs_mass_frac_↩ from_parent1, double total_mass1)

  This is a class to create a child parent instance.

Public Attributes

- std::string **name**
- double **childs_mass_frac_from_parent**
- double **parents_mass_frac_to_child**
- double **total_mass**

4.1.1 Constructor & Destructor Documentation

4.1.1.1 ChildParentInstance()

```
ChildParentInstance::ChildParentInstance (
            std::string name_arg,
            double parents_mass_frac_to_child_arg,
            double childs_mass_frac_from_parent_arg,
            double total_mass_arg)
```

This is a class to create a child parent instance.

Parameters

name_arg	The name of the child parent instance
parents_mass_frac_to_child_arg	The mass fraction from parent to child
childs_mass_frac_from_parent_arg	The mass fraction from child to parent
total_mass_arg	The total mass of the child parent instance

Returns

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

- include/matcher/structs_and_classes.h
- src/matcher/utils/structs_and_classes.cpp

4.2 CitySnaps Class Reference

Public Member Functions

- CitySnaps (int snapnums1, int snapnums2)

  This is a class to create a city snaps instance. A city contains member clouds from two snapshots being matched/compared.

Public Attributes

- std::string **snap_name1**
- std::string **snap_name2**
- int **snap_num1**
- int **snap_num2**
- CloudGroup **parent_group**
- CloudGroup **child_group**

4.2.1 Constructor & Destructor Documentation

4.2.1.1 CitySnaps()

```
CitySnaps::CitySnaps (
            int snapnums1,
            int snapnums2)
```

This is a class to create a city snaps instance. A city contains member clouds from two snapshots being matched/compared.

Parameters

snapnums1	The first snapshot number
snapnums2	The second snapshot number

Returns

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

- include/matcher/structs_and_classes.h
- src/matcher/utils/structs_and_classes.cpp

4.3 CloudGroup Struct Reference

Public Member Functions

- void add_member (MemberCloud cloud)

  This is a class to add a member cloud to a cloud group.

Public Attributes

- std::string **snap_name**
- int **snap_num**
- std::vector< MemberCloud > **members**

4.3.1 Member Function Documentation

4.3.1.1 add_member()

```
void CloudGroup::add_member (
            MemberCloud cloud)
```

This is a class to add a member cloud to a cloud group.

Parameters

cloud	The member cloud to be added

Returns

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

- include/matcher/structs_and_classes.h
- src/matcher/utils/structs_and_classes.cpp

4.4 CloudStruct Struct Reference

Public Attributes

- std::string **name**
- int **num**
- int **snap**
- double **total_mass**
- std::string **final_name**
- int **end_of_gen**
- int **flag**

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

- include/linker/structs_and_classes.h

4.5 Group_struct Struct Reference

Public Attributes

- std::string **name**
- Group_struct * **subgroup**

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

- include/linker/structs_and_classes.h
- include/matcher/structs_and_classes.h

4.6 MemberCloud Class Reference

Public Member Functions

- MemberCloud (std::string name_arg, std::vector< double > pIDs, std::vector< double > pIDgen, std::vector< double > masses_arg)

  This is a class to create a member cloud.
- void add_child (std::string name, double parents_mass_frac_to_child, double childs_mass_frac_from_parent, double total_child_mass)

  This is a class which contains information about clouds in a city (combination of two snapshots)
- void add_parent (std::string name, double parents_mass_frac_to_child, double childs_mass_frac_from_parent, double total_parent_mass)

  This is a class to add a parent to a member cloud.

Public Attributes

- std::string **name**
- std::vector< double > **particleIDs**
- std::vector< double > **particleIDgens**
- std::vector< double > **masses**
- std::vector< double > **dummy_pIDs**
- std::vector< double > **dummy_masses**
- double **total_mass**
- std::vector< ChildParentInstance > **children**
- std::vector< ChildParentInstance > **parents**
- int **num_children**
- int **num_parents**

4.6.1 Constructor & Destructor Documentation

4.6.1.1 MemberCloud()

```
MemberCloud::MemberCloud (
            std::string name_arg,
            std::vector< double > pIDs,
            std::vector< double > pIDgen,
            std::vector< double > masses_arg)
```

This is a class to create a member cloud.

Parameters

name_arg	The name of the member cloud
pIDs	The particle IDs of the member cloud
pIDgen	The particle ID generator of the member cloud
masses_arg	The masses of the member cloud

Returns

4.6.2 Member Function Documentation

4.6.2.1 add_child()

```
void MemberCloud::add_child (
            std::string name,
            double parents_mass_frac_to_child,
            double childs_mass_frac_from_parent,
            double total_child_mass)
```

This is a class which contains information about clouds in a city (combination of two snapshots)

Parameters

name	The name of the child
parents_mass_frac_to_child	The mass fraction from parent to child
childs_mass_frac_from_parent	The mass fraction from child to parent
total_child_mass	The total mass of the child

Returns

4.6.2.2 add_parent()

```
void MemberCloud::add_parent (
            std::string name,
            double parents_mass_frac_to_child,
            double childs_mass_frac_from_parent,
            double total_parent_mass)
```

This is a class to add a parent to a member cloud.

Parameters

name	The name of the parent
parents_mass_frac_to_child	The mass fraction from parent to child
childs_mass_frac_from_parent	The mass fraction from child to parent
total_parent_mass	The total mass of the parent

Returns

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

- include/matcher/structs_and_classes.h
- src/matcher/utils/structs_and_classes.cpp

4.7 Params Struct Reference

Public Attributes

- int **last_snap**
- int **first_snap**
- std::string **cloud_prefix**
- std::string **path**
- std::string **filename_base_prefix**
- std::string **filename_base_suffix**
- std::string **dat_filename_base_prefix**
- std::string **dat_filename_base_suffix**
- std::string **write_filename_base_prefix**
- std::string **write_filename_base_suffix**
- std::string **linker_output_filename_prefix**
- std::string **file_arch_root**
- std::string **file_arch_cloud_subgroup**
- std::string **file_arch_masses_field**
- std::string **file_arch_pIDs_field**
- int **particle_lower_limit**
- double **threshold_frac_for_child**
- std::string **file_arch_pIDgen_field**

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

- include/linker/structs_and_classes.h
- include/matcher/structs_and_classes.h

Chapter 5

File Documentation

5.1 io.h

```
00001 #ifndef IO_H
00002 #define IO_H
00003
00004
00005 #include <hdf5.h>
00006 #include <vector>
00007 #include <iostream>
00008 #include <string>
00009
00010 #include "structs_and_classes.h"
00011 #include "linker_functions.h"
00012 #include "utilities.h"
00013 #include "read_params.h"
00014
00015
00016 hid_t get_last_group(Group_struct *subgroup_struct, hid_t group);
00017 std::vector<double> read_cloud_data_double(Params &params, int snap_num,
00018                     std::string field_to_read, std::string cloud_name);
00019 double read_mass(Params &params, int snap_num, std::string cloud_name);
00020 herr_t child_info(hid_t loc_id, const char *name, const H5L_info_t *linfo, void *opdata);
00021 void get_child_list(Params &params, int snap_num, std::string cloud_name,
00022                     std::vector<std::string> &child_list_names,
00023                     std::vector<double> &child_list_fracs);
00024 void print_list(std::vector<std::vector<CloudStruct» &tracked_cloud_list);
00025 void write_to_file(std::vector<std::vector<CloudStruct» &tracked_cloud_list, Params &params,
     std::string field_name);
00026
00027
00028
00029 #endif // IO_H
```

5.2 io.h

```
00001 #ifndef IO_H
00002 #define IO_H
00003
00004 #include "structs_and_classes.h"
00005 #include "matcher_functions.h"
00006 #include "utilities.h"
00007 #include <vector>
00008 #include <fstream>
00009 #include <hdf5.h>
00010
00011 void write_to_hdf5_file(CitySnaps& snapsnap, Params& params, int parent_num_clouds, int
     child_num_clouds);
00012 //void write_to_hdf5_file(CitySnaps snapsnap, Params &params, int parent_num_clouds, int
     child_num_clouds);
00013 void writeCloudData(hid_t group_id, const MemberCloud& member, int child_flag);
00014 void writeGroupDataChildren(hid_t group_id, const std::string& subgroup_name, const MemberCloud&
     member);
00015 void writeGroupDataParents(hid_t group_id, const std::string& subgroup_name, const MemberCloud&
     member);
00016 hid_t get_last_group(Group_struct* subgroup_struct, hid_t group_id);
```

```
00017 std::vector<int> read_cloud_data_int(Params& params, int snap_num, const std::string& field_to_read,
      const std::string& cloud_name);
00018 std::vector<double> read_cloud_data_double(Params& params, int snap_num, std::string& field_to_read,
      std::string& cloud_name);
00019
00020 #endif
```

5.3 linker_functions.h

```
00001 #ifndef LINKER_FUNCTIONS_H
00002 #define LINKER_FUNCTIONS_H
00003
00004 #include <hdf5.h>
00005 #include <vector>
00006 #include <iostream>
00007 #include <string>
00008 #include <algorithm>
00009 #include <cmath>
00010 #include <cstring>
00011 #include <fstream>
00012 #include "structs_and_classes.h"
00013 #include "read_params.h"
00014
00015
00016 void load_to_cloudlist(std::vector<CloudStruct> &cloud_list, CloudStruct &cloud);
00017 void load_to_tracked_cloud_list(std::vector<std::vector<CloudStruct>> &tracked_cloud_list, CloudStruct
      &cloud, std::string key, int index);
00018 int find_proper_child(std::vector<std::string> &child_list_names, std::vector<double>
      &child_list_fracs, std::vector<std::string> &child_list_names_original, std::vector<CloudStruct>
      &cloud_list, int index_to_omit, int snap_num, Params &params);
00019 int check_if_cloud_exists_in_list(CloudStruct &cloud, std::vector<CloudStruct> &cloud_list);
00020 void linker(Params &params);
00021
00022 #endif
```

5.4 read_params.h

```
00001 #ifndef READ_PARAMS_H
00002 #define READ_PARAMS_H
00003
00004 #include "structs_and_classes.h"
00005
00006 #include <iostream>
00007 #include <fstream>
00008 #include <sstream>
00009 #include <string>
00010
00011
00012 void trim(std::string& s);
00013 void printParams(const Params& params);
00014 bool parseParams(const std::string& filename, Params& params, const std::string& name, const
      std::string& sim_name);
00015
00016 #endif
```

5.5 read_params.h

```
00001 #ifndef READ_PARAMS_H
00002 #define READ_PARAMS_H
00003
00004 #include "structs_and_classes.h"
00005 //#include "matcher_functions.h"
00006
00007 #include <iostream>
00008 #include <fstream>
00009 #include <sstream>
00010 #include <string>
00011
00012
00013 void trim(std::string& s);
00014 void printParams(const Params& params);
00015 bool parseParams(const std::string& filename, Params& params, const std::string& name, const
      std::string& sim_name);
00016
00017 #endif
```

5.6 structs_and_classes.h

```
00001 #ifndef STRUCTS_AND_CLASSES_H
00002 #define STRUCTS_AND_CLASSES_H
00003
00004 #include <string>
00005
00006 struct Params{
00007     int last_snap, first_snap;
00008         std::string cloud_prefix, path, filename_base_prefix, filename_base_suffix,
      dat_filename_base_prefix, dat_filename_base_suffix;
00009         std::string write_filename_base_prefix, write_filename_base_suffix,
      linker_output_filename_prefix;
00010         std::string file_arch_root, file_arch_cloud_subgroup, file_arch_masses_field,
      file_arch_pIDs_field;
00011         int particle_lower_limit; double threshold_frac_for_child;
00012 };
00013
00014 struct Group_struct{
00015         std::string name;
00016         Group_struct *subgroup;
00017 };
00018 Group_struct* create_group(std::string name);
00019
00020 struct CloudStruct{
00021         std::string name;
00022         int num, snap; double total_mass;
00023         std::string final_name;
00024         int end_of_gen;
00025         int flag;                       // 0 if not added yet. 1 if added.
00026 };
00027 CloudStruct initialize_cloud(int snap_num, std::string cloud_name, Params &params);
00028
00029 #endif
```

5.7 include/matcher/structs_and_classes.h File Reference

Contains the definitions of the structs and classes used in the CloudTracker program.

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

Classes

- struct Group_struct
- struct Params
- class ChildParentInstance
- class MemberCloud
- struct CloudGroup
- class CitySnaps

Functions

- Group_struct * create_group (std::string name)

  Creates a group struct with the given name.

5.7.1 Detailed Description

Contains the definitions of the structs and classes used in the CloudTracker program.

Author

Shivan Khullar

Date

June 2024

5.7.2 Function Documentation

5.7.2.1 create_group()

```
Group_struct * create_group (
            std::string name)
```

Creates a group struct with the given name.

Parameters

name	The name of the group.

Returns

A pointer to the created group struct.

Creates a group struct with the given name.

Parameters

name	The name of the group to be created

Returns

5.8 structs_and_classes.h

Go to the documentation of this file.

```
00001
00005
00006
00007
00008 #ifndef STRUCTS_AND_CLASSES_H
00009 #define STRUCTS_AND_CLASSES_H
00010
00011 #include <iostream>
00012 #include <string>
00013 //#include <rarray>
00014 #include <vector>
00015
00016 //#include <H5Cpp.h>
00017 //using namespace H5;
00018
00019 //int check_if_exists(std::vector<double> a, double b);
00020 //int get_first_index(std::vector<double> a, double b);
00021
00022
00023 struct Group_struct{
00024         std::string name;
00025         Group_struct *subgroup;
00026 };
00027 Group_struct* create_group(std::string name);
00028
00029
00030 struct Params{
00031         int last_snap, first_snap;
00032         std::string cloud_prefix, path, filename_base_prefix, filename_base_suffix,
      dat_filename_base_prefix, dat_filename_base_suffix;
00033     std::string write_filename_base_prefix, write_filename_base_suffix;
00034     std::string file_arch_root, file_arch_cloud_subgroup, file_arch_masses_field,
      file_arch_pIDs_field, file_arch_pIDgen_field;
00035     int particle_lower_limit;
00036 };
00037
00038 class ChildParentInstance{
00039 public:
00040         std::string name;
00041     double childs_mass_frac_from_parent;
00042     double parents_mass_frac_to_child;
00043     double total_mass;
00044         ChildParentInstance(std::string name1, double parents_mass_frac_to_child1, double
      childs_mass_frac_from_parent1, double total_mass1);
00045 };
00046
00047 class MemberCloud{
00048 public:
00049         std::string name;
00050         std::vector<double> particleIDs;
00051         std::vector<double> particleIDgens;
00052         std::vector<double> masses;
00053     std::vector<double> dummy_pIDs;
00054     std::vector<double> dummy_masses;
00055         double total_mass;
00056         std::vector<ChildParentInstance> children;
00057         std::vector<ChildParentInstance> parents;
00058     int num_children, num_parents;
00059     MemberCloud(std::string name_arg, std::vector<double> pIDs, std::vector<double> pIDgen,
      std::vector<double> masses_arg);
00060     void add_child(std::string name, double parents_mass_frac_to_child, double
      childs_mass_frac_from_parent, double total_child_mass);
00061         void add_parent(std::string name, double parents_mass_frac_to_child, double
      childs_mass_frac_from_parent, double total_parent_mass);
00062 };
00063
00064 struct CloudGroup{
00065         std::string snap_name;
00066         int snap_num;
00067         std::vector<MemberCloud> members;
00068         void add_member(MemberCloud cloud);
00069 };
00070
00071 class CitySnaps{
00072 public:
00073         std::string snap_name1, snap_name2;
00074         int snap_num1, snap_num2;
00075         CloudGroup parent_group, child_group;
00076         CitySnaps(int snapnums1, int snapnums2);
00077 };
00078
00079
00080 #endif
```

5.9 utilities.h

```
00001 #ifndef UTILITIES_H
00002 #define UTILITIES_H
00003
00004 #include <hdf5.h>
00005 #include <vector>
00006 #include <iostream>
00007 #include <string>
00008 #include <algorithm>
00009 #include <cmath>
00010 #include <cstring>
00011 #include <fstream>
00012 #include "structs_and_classes.h"
00013 #include "read_params.h"
00014
00015
00016 std::string get_snapshot_name(int i);
00017 std::string get_cloud_name(int i, Params &params);
00018 int find_num_clouds(int snap_num, Params &params);
00019 int check_if_exists(std::vector<double> a, double b);
00020 int get_num_particles(Params &params, int snap_num, std::string cloud_name);
00021 void print_list(std::vector<std::vector<CloudStruct» &tracked_cloud_list);
00022 void write_to_file(std::vector<std::vector<CloudStruct» &tracked_cloud_list, Params &params,
      std::string field_name);
00023 void linker(Params &params);
00024
00025 #endif // UTILITIES_H
```

5.10 utilities.h

```
00001
00002
00003 #ifndef UTILITIES_H
00004 #define UTILITIES_H
00005
00006 #include "structs_and_classes.h"
00007 #include <vector>
00008
00009 int check_if_value_exists_in_array(std::vector<double> a, double b);
00010 int get_first_index(std::vector<double> a, double b);
00011 void print_array_double(std::vector<double> a);
00012 void print_array_int(std::vector<int> a);
00013 double sum_array(std::vector<double> a);
00014 std::string get_snapshot_name(int i);
00015 int find_num_clouds(int snap_num, Params &params);
00016
00017
00018 #endif
```

5.11 matcher_functions.h

```
00001 #ifndef MATCHER_FUNCTIONS_H
00002 #define MATCHER_FUNCTIONS_H
00003
00004 #include "structs_and_classes.h"
00005 #include "read_params.h"
00006 #include "io.h"
00007 #include "utilities.h"
00008 #include <iostream>
00009 #include <string>
00010 #include <vector>
00011 #include <fstream>
00012 //#include <H5Cpp.h>
00013 //using namespace H5;
00014
00015 //void write_to_hdf5_file(CitySnaps snapsnap, Params &params, int parent_num_clouds, int
      child_num_clouds);
00016 //H5::Group get_last_group(Group_struct *subgroup_struct, H5::Group group);
00017 //std::string get_snapshot_name(int i);
00018 //std::vector<int> read_cloud_data_int(Params &params, int snap_num, std::string field_to_read,
      std::string cloud_name);
00019 //std::vector<double> read_cloud_data_double(Params &params, int snap_num, std::string field_to_read,
      std::string cloud_name);
00020 //double sum_array(std::vector<double> a);
00021 int compare_particles(std::vector<int> a, std::vector<int> b);
00022 double mass_frac(std::vector<int> parent_pIDs, std::vector<int> child_pIDs, std::vector<double>
      parent_masses, std::vector<double> child_masses,
00023                  int num_common_particles);
```

```
00024 std::string get_cloud_name(int i, Params &params);
00025 void load_clouds_to_group(int snap_num, int num_clouds_parent_snap, Params &params, CitySnaps
      &snapsnap, std::string key);
00026 //int find_num_clouds(int snap_num, Params &params);
00027 void matcher(Params &params);
00028
00029 #endif
```

5.12 src/linker/io/io_hdf5.cpp File Reference

Contains functions to read data from HDF5 files.

```
#include <hdf5.h>
#include "../../../include/linker/io.h"
```

Functions

- hid_t get_last_group (Group_struct *subgroup_struct, hid_t group)

  Function to get the last group (traverse down the hdf5 file structure to the dataset)
- std::vector< double > read_cloud_data_double (Params &params, int snap_num, std::string field_to_read, std::string cloud_name)

  Function to read double data from an hdf5 file.
- double read_mass (Params &params, int snap_num, std::string cloud_name)

  Function to read the mass of a cloud from the hdf5 file generated by the matcher program.
- herr_t child_info (hid_t loc_id, const char *name, const H5L_info_t *linfo, void *opdata)

  Function to get the information of a child (used in get_child_list to get the names of and mass donated/obtained fractions of the children.
- void get_child_list (Params &params, int snap_num, std::string cloud_name, std::vector< std::string > &child_list_names, std::vector< double > &child_list_fracs)

  Function to get the list of children of a cloud.

5.12.1 Detailed Description

Contains functions to read data from HDF5 files.

Author

Shivan Khullar

Date

June 2024

5.12.2 Function Documentation

5.12.2.1 child_info()

```
herr_t child_info (
            hid_t loc_id,
            const char * name,
            const H5L_info_t * linfo,
            void * opdata)
```

Function to get the information of a child (used in get_child_list to get the names of and mass donated/obtained fractions of the children.

Parameters

loc_id	The location id
name	The name of the child
linfo	The link info
opdata	The operation data

Returns

The error code

5.12.2.2 get_child_list()

```
void get_child_list (
            Params & params,
            int snap_num,
            std::string cloud_name,
            std::vector< std::string > & child_list_names,
            std::vector< double > & child_list_fracs)
```

Function to get the list of children of a cloud.

Parameters

params	The parameters struct
snap_num	The snapshot number
cloud_name	The name of the cloud
child_list_names	The list of names of the children
child_list_fracs	The list of mass fractions donated by the parent, to the children

5.12.2.3 get_last_group()

```
hid_t get_last_group (
            Group_struct * subgroup_struct,
            hid_t group)
```

Function to get the last group (traverse down the hdf5 file structure to the dataset)

Parameters

subgroup_struct	The subgroup instance of a group struct
group	Name of the group or subgroup

Returns

5.12.2.4 read_cloud_data_double()

```
std::vector< double > read_cloud_data_double (
            Params & params,
            int snap_num,
            std::string field_to_read,
            std::string cloud_name)
```

Function to read double data from an hdf5 file.

Parameters

params	The parameters struct
snap_num	The snapshot number
field_to_read	The field to read from the dataset
cloud_name	The name of the cloud

Returns

The data read from the dataset

5.12.2.5 read_mass()

```
double read_mass (
            Params & params,
            int snap_num,
            std::string cloud_name)
```

Function to read the mass of a cloud from the hdf5 file generated by the matcher program.

Parameters

params	The parameters struct
snap_num	The snapshot number
cloud_name	The name of the cloud

Returns

The mass of the cloud

5.13 src/linker/io/io_txt.cpp File Reference

Contains functions to read and write data to text files. All the tracked cloud list info is stored in text files. Linker generates two text files, one for the names of the clouds and the other for the masses of the clouds.

```
#include "../../../include/linker/io.h"
#include "../../../include/linker/read_params.h"
#include "../../../include/linker/structs_and_classes.h"
#include "../../../include/linker/utilities.h"
#include <fstream>
#include <iostream>
#include <string>
#include <vector>
```

Functions

- void print_list (std::vector< std::vector< CloudStruct > > &tracked_cloud_list)

  Function to print the tracked cloud list.
- void write_to_file (std::vector< std::vector< CloudStruct > > &tracked_cloud_list, Params &params, std::string field_name)

  Function to write the tracked cloud list to a text file.

5.13.1 Detailed Description

Contains functions to read and write data to text files. All the tracked cloud list info is stored in text files. Linker generates two text files, one for the names of the clouds and the other for the masses of the clouds.

Author

Shivan Khullar

Date

June 2024

5.13.2 Function Documentation

5.13.2.1 print_list()

```
void print_list (
            std::vector< std::vector< CloudStruct > > & tracked_cloud_list)
```

Function to print the tracked cloud list.

Parameters

tracked_cloud_list	The list containing the tracked clouds

5.13.2.2 write_to_file()

```
void write_to_file (
            std::vector< std::vector< CloudStruct > > & tracked_cloud_list,
            Params & params,
            std::string field_name)
```

Function to write the tracked cloud list to a text file.

Parameters

tracked_cloud_list	The list containing the tracked clouds
params	The parameters struct
field_name	The field name to write to the file, names or masses

5.14 src/linker/io/read_params.cpp File Reference

Contains functions to read parameters from a file The parameters are stored in a text file and read into a struct Linker then reads these parameters to perform the linking process.

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <string>
#include "../../../include/linker/read_params.h"
```

Functions

- void trim (std::string &s)

  Function to remove leading and trailing whitespaces from a string.
- void printParams (const Params &params)

  Function to print the parameters.
- bool parseParams (const std::string &filename, Params &params, const std::string &name, const std::string &sim_name)

  Function to parse the parameters from a file.

5.14.1 Detailed Description

Contains functions to read parameters from a file The parameters are stored in a text file and read into a struct Linker then reads these parameters to perform the linking process.

Author

Shivan Khullar

Date

June 2024

5.14.2 Function Documentation

5.14.2.1 parseParams()

```
bool parseParams (
            const std::string & filename,
            Params & params,
            const std::string & name,
            const std::string & sim_name)
```

Function to parse the parameters from a file.

Parameters

filename	The name of the file containing the parameters
params	The parameters struct
name	The name of the parameter
sim_name	The name of the simulation

Returns

True if the parameters are parsed successfully, false otherwise

5.14.2.2 printParams()

```
void printParams (
            const Params & params)
```

Function to print the parameters.

Parameters

params	The parameters struct

5.14.2.3 trim()

```
void trim (
            std::string & s)
```

Function to remove leading and trailing whitespaces from a string.

Parameters

s	The string to trim

5.15 src/matcher/io/read_params.cpp File Reference

This file contains the implementation of the functions that read in the parameters from the parameter file. The parameters are stored in a struct called Params. The functions are used in the main driver to read in the parameters and assign them to the code variables.

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <string>
#include "../../../include/matcher/read_params.h"
```

Functions

- void trim (std::string &s)

  Trims the leading and trailing whitespaces from a string.
- void printParams (const Params &params)

  Prints the parameters read from the parameter file to the console.
- bool parseParams (const std::string &filename, Params &params, const std::string &name, const std::string &sim_name)

  Parses the parameters from the parameter file and assigns them to the Params struct.

5.15.1 Detailed Description

This file contains the implementation of the functions that read in the parameters from the parameter file. The parameters are stored in a struct called Params. The functions are used in the main driver to read in the parameters and assign them to the code variables.

Author

Shivan Khullar

Date

June 2024

5.15.2 Function Documentation

5.15.2.1 parseParams()

```
bool parseParams (
            const std::string & filename,
            Params & params,
            const std::string & name,
            const std::string & sim_name)
```

Parses the parameters from the parameter file and assigns them to the Params struct.

Function to parse the parameters from a file.

Parameters

filename	path to the parameter file
params	the struct to store the parameters
name	the name of the identification configuration (e.g. - n10_alpha2 for CloudPhinder)
sim_name	the name of the simulation

Returns

True if successful, false if not

5.15.2.2 printParams()

```
void printParams (
            const Params & params)
```

Prints the parameters read from the parameter file to the console.

Function to print the parameters.

Parameters

params	

5.15.2.3 trim()

```
void trim (
            std::string & s)
```

Trims the leading and trailing whitespaces from a string.

Function to remove leading and trailing whitespaces from a string.

Parameters

s	

5.16 src/linker/main/linker.cpp File Reference

Main driver for the linker part of the code which links clouds in different snapshots together.

```
#include <hdf5.h>
#include <vector>
#include <iostream>
#include <string>
#include <algorithm>
#include <cmath>
#include <cstring>
#include <fstream>
#include "../../../include/linker/linker_functions.h"
#include "../../../include/linker/structs_and_classes.h"
#include "../../../include/linker/read_params.h"
#include "../../../include/linker/utilities.h"
#include "../../../include/linker/io.h"
```

Functions

- int main (int argc, char ∗argv[ ])
  Main driver function.

5.16.1 Detailed Description

Main driver for the linker part of the code which links clouds in different snapshots together.

Author

Shivan Khullar

Date

June 2024

5.16.2 Function Documentation

5.16.2.1 main()

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

Main driver function.

Parameters

argc	number of command line arguments
argv	command line arguments as strings

Returns

5.17 src/linker/main/linker_functions.cpp File Reference

Contains the functions used in linking the clouds together.

```
#include "../../../include/linker/linker_functions.h"
#include "../../../include/linker/read_params.h"
#include "../../../include/linker/structs_and_classes.h"
#include "../../../include/linker/utilities.h"
#include "../../../include/linker/io.h"
```

Functions

- void load_to_cloudlist (std::vector< CloudStruct > &cloud_list, CloudStruct &cloud)

  Loads a cloud into the cloudlist (the cloudlist is a list of all clouds, not just tracked ones)
- void load_to_tracked_cloud_list (std::vector< std::vector< CloudStruct > > &tracked_cloud_list, CloudStruct &cloud, std::string key, int index)

  Loads a cloud into the tracked cloud list.
- int find_proper_child (std::vector< std::string > &child_list_names, std::vector< double > &child_list_fracs, std::vector< std::string > &child_list_names_original, std::vector< CloudStruct > &cloud_list, int index_to↩_omit, int snap_num, Params &params)

  This function finds the "proper child" of any cloud X.
- int check_if_cloud_exists_in_list (CloudStruct &cloud, std::vector< CloudStruct > &cloud_list)

  This is a function to check if a cloud already exists in the cloud list.
- void linker (Params &params)

  This function links all the clouds together.

5.17.1 Detailed Description

Contains the functions used in linking the clouds together.

Author

Shivan Khullar

Date

June 2024

5.17.2 Function Documentation

5.17.2.1 check_if_cloud_exists_in_list()

```
int check_if_cloud_exists_in_list (
            CloudStruct & cloud,
            std::vector< CloudStruct > & cloud_list)
```

This is a function to check if a cloud already exists in the cloud list.

Parameters

cloud	The cloud to check for
cloud_list	The cloud list of all clouds to check in

Returns

Return 1 if found

5.17.2.2 find_proper_child()

```
int find_proper_child (
            std::vector< std::string > & child_list_names,
            std::vector< double > & child_list_fracs,
            std::vector< std::string > & child_list_names_original,
            std::vector< CloudStruct > & cloud_list,
            int index_to_omit,
            int snap_num,
            Params & params)
```

This function finds the "proper child" of any cloud X.

Parameters

child_list_names	This is a list of all clouds that are children of the cloud X
child_list_fracs	This is a list of all the fractions of mass child clouds have derived from the parent cloud

child_list_names_original	This is the original list of child cloud names, needed because this function is recursive and we delete the children if they have already been added in the cloud list (not the tracked cloud list, the other one)
cloud_list	This is the list of all clouds (not just the tracked clouds)
index_to_omit	If there is a cloud in the child list to omit. Used in recurrent calls to ignore a child of a more massive parent if it is the proper child
snap_num	The snapshot number of the cloud X
params	The params class object. Needed because we have to

Returns

Return the index of the proper child amongst all the child clouds of cloud X.

5.17.2.3 linker()

```
void linker (
            Params & params)
```

This function links all the clouds together.

Parameters

The	parameter struct which contains meta info.

Returns

5.17.2.4 load_to_cloudlist()

```
void load_to_cloudlist (
            std::vector< CloudStruct > & cloud_list,
            CloudStruct & cloud)
```

Loads a cloud into the cloudlist (the cloudlist is a list of all clouds, not just tracked ones)

Parameters

cloud_list	The list of cloud objects
cloud	The cloud object to be added to the list

5.17.2.5 load_to_tracked_cloud_list()

```
void load_to_tracked_cloud_list (
            std::vector< std::vector< CloudStruct > > & tracked_cloud_list,
            CloudStruct & cloud,
            std::string key,
            int index)
```

Loads a cloud into the tracked cloud list.

Parameters

tracked_cloud_list	The list of tracked cloud objects
cloud	The cloud to be added to the list
key	Key to signify if this cloud is part of a chain
index	This specifies which generation the cloud belongs to

5.18 src/linker/utils/structs_and_classes.cpp File Reference

Contains the structs and classes used in the linker program.

```
#include "../../../include/linker/structs_and_classes.h"
#include "../../../include/linker/linker_functions.h"
#include "../../../include/linker/utilities.h"
```

Functions

- Group_struct * create_group (std::string name)
  Creates a group struct with the given name.
- CloudStruct initialize_cloud (int snap_num, std::string cloud_name, Params &params)
  Initializes a cloud struct with the given snapshot number and cloud name.

5.18.1 Detailed Description

Contains the structs and classes used in the linker program.

Author

Shivan Khullar

Date

June 2024

5.18.2 Function Documentation

5.18.2.1 create_group()

```
Group_struct * create_group (
            std::string name)
```

Creates a group struct with the given name.

Parameters

name	The name of the group.

Returns

A pointer to the created group struct.

5.18.2.2 initialize_cloud()

```
CloudStruct initialize_cloud (
            int snap_num,
            std::string cloud_name,
            Params & params)
```

Initializes a cloud struct with the given snapshot number and cloud name.

Parameters

snap_num	The snapshot number of the cloud.
cloud_name	The name of the cloud.
params	The parameters struct.

Returns

The initialized cloud struct.

5.19 src/matcher/utils/structs_and_classes.cpp File Reference

This file contains the definitions of the classes and structures used in the matcher application. This includes the Group_struct, Params, ChildParentInstance, MemberCloud, and CitySnaps classes. The Group_struct class is used to read in the group structure from the HDF5 file. The ChildParentInstance class is used to store information about a child cloud and its parent.
The MemberCloud class is used to store information about a cloud entity in a snapshot. The CitySnaps class is used to store information about clouds in a city (combination of two snapshots). The CloudGroup class is a group of clouds in a city.

```
#include <string>
#include "../../../include/matcher/structs_and_classes.h"
#include "../../../include/matcher/utilities.h"
#include "../../../include/matcher/matcher_functions.h"
```

Functions

- Group_struct * create_group (std::string name)

  This is a class to create a group structure It is used to read in the group structure and group substructure from the HDF5 file.

5.19.1 Detailed Description

This file contains the definitions of the classes and structures used in the matcher application. This includes the Group_struct, Params, ChildParentInstance, MemberCloud, and CitySnaps classes. The Group_struct class is used to read in the group structure from the HDF5 file. The ChildParentInstance class is used to store information about a child cloud and its parent.
The MemberCloud class is used to store information about a cloud entity in a snapshot. The CitySnaps class is used to store information about clouds in a city (combination of two snapshots). The CloudGroup class is a group of clouds in a city.

Author

Shivan Khullar

Date

June 2024

5.19.2 Function Documentation

5.19.2.1 create_group()

```
Group_struct * create_group (
            std::string name)
```

This is a class to create a group structure It is used to read in the group structure and group substructure from the HDF5 file.

Creates a group struct with the given name.

Parameters

name	The name of the group to be created

Returns

5.20 src/linker/utils/utilities.cpp File Reference

Contains some generic utilities like finding the number of clouds in a snapshot, getting the name of the snapshot or getting the name of the cloud if you have a number. Also contains functions for checking two arrays and seeing whether they have something in common, and getting the number of particles in a cloud.

```
#include "../../../include/linker/utilities.h"
#include "../../../include/linker/io.h"
#include "../../../include/linker/read_params.h"
```

Functions

- std::string get_snapshot_name (int i)

  Function to obtain the snapshot name by adding required number of zeros.
- std::string get_cloud_name (int i, Params &params)

  Function to get the cloud name by adding required number of zeros.
- int find_num_clouds (int snap_num, Params &params)

  Function to find the number of clouds in a snapshot.
- int check_if_exists (std::vector< double > a, double b)

  Function to check if a number exists in a vector.
- int get_num_particles (Params &params, int snap_num, std::string cloud_name)

  Function to get the number of particles in a cloud.

5.20.1 Detailed Description

Contains some generic utilities like finding the number of clouds in a snapshot, getting the name of the snapshot or getting the name of the cloud if you have a number. Also contains functions for checking two arrays and seeing whether they have something in common, and getting the number of particles in a cloud.

Author

Shivan Khullar

Date

June 2024

5.20.2 Function Documentation

5.20.2.1 check_if_exists()

```
int check_if_exists (
            std::vector< double > a,
            double b)
```

Function to check if a number exists in a vector.

Parameters

a	The vector
b	The number to check

Returns

The number of times the number exists in the vector

5.20.2.2 find_num_clouds()

```
int find_num_clouds (
            int snap_num,
            Params & params)
```

Function to find the number of clouds in a snapshot.

Parameters

snap_num	The snapshot number
params	The parameters struct

Returns

The number of clouds in the snapshot

5.20.2.3 get_cloud_name()

```
std::string get_cloud_name (
            int i,
            Params & params)
```

Function to get the cloud name by adding required number of zeros.

Parameters

i	The cloud number
params	The parameters struct

Returns

The cloud name

5.20.2.4 get_num_particles()

```
int get_num_particles (
            Params & params,
            int snap_num,
            std::string cloud_name)
```

Function to get the number of particles in a cloud.

Parameters

params	The parameters struct
snap_num	The snapshot number
cloud_name	The cloud name

Returns

The number of particles in the cloud

5.20.2.5 get_snapshot_name()

```
std::string get_snapshot_name (
            int i)
```

Function to obtain the snapshot name by adding required number of zeros.

Parameters

i	The snapshot number

Returns

5.21 src/matcher/utils/utilities.cpp File Reference

This file contains utility functions for the matcher application. This includes functions to check if a double exists in a vector, get the first index of a double in a vector, print a double array, print an int array, sum the elements of a double array, get the name of the snapshot part of a Cloud, find the number of clouds in a snapshot, and get the name of a snapshot as a string with leading zeros to make it a 3-digit number. If you are working with data that is not the default structure, you will have to alter the get_snapshot_name() and find_num_clouds() functions.

```
#include <string>
#include "../../../include/matcher/structs_and_classes.h"
#include "../../../include/matcher/matcher_functions.h"
```

Functions

- int check_if_value_exists_in_array (std::vector< double > a, double b)

  Check if a double exists in a vector.
- int get_first_index (std::vector< double > a, double b)

  Get the first index of a double in a vector.
- void print_array_double (std::vector< double > a)

  Function to print a double array.
- void print_array_int (std::vector< int > a)

  Function to print an int array.
- double sum_array (std::vector< double > a)

  Function to sum the elements of a double array.
- std::string get_snapshot_name (int i)

  Function to get the name of the snapshot part of a Cloud.
- int find_num_clouds (int snap_num, Params &params)

  Function to find the number of clouds in a snapshot.

5.21.1 Detailed Description

This file contains utility functions for the matcher application. This includes functions to check if a double exists in a vector, get the first index of a double in a vector, print a double array, print an int array, sum the elements of a double array, get the name of the snapshot part of a Cloud, find the number of clouds in a snapshot, and get the name of a snapshot as a string with leading zeros to make it a 3-digit number. If you are working with data that is not the default structure, you will have to alter the get_snapshot_name() and find_num_clouds() functions.

Author

Shivan Khullar

Date

June 2024

5.21.2 Function Documentation

5.21.2.1 check_if_value_exists_in_array()

```
int check_if_value_exists_in_array (
            std::vector< double > a,
            double b)
```

Check if a double exists in a vector.

Parameters

a	The vector to check
b	The double to check for

5.21.2.2 find_num_clouds()

```
int find_num_clouds (
            int snap_num,
            Params & params)
```

Function to find the number of clouds in a snapshot.

Parameters

snap_num	: the snapshot number
params	: the parameters for matcher application (read from the .txt file in main/matcher.cpp)

Returns

the number of clouds in the snapshot

5.21.2.3 get_first_index()

```
int get_first_index (
            std::vector< double > a,
            double b)
```

Get the first index of a double in a vector.

Parameters

a	The vector to check
b	The double to check for

5.21.2.4 get_snapshot_name()

```
std::string get_snapshot_name (
            int i)
```

Function to get the name of the snapshot part of a Cloud.

Function to obtain the snapshot name by adding required number of zeros.

Parameters

i	: the snapshot number

Returns

the name of a snapshot as a string, with leading zeros to make it a 3-digit number

5.21.2.5 print_array_double()

```
void print_array_double (
            std::vector< double > a)
```

Function to print a double array.

Parameters

a	: the array to be printed

5.21.2.6 print_array_int()

```
void print_array_int (
            std::vector< int > a)
```

Function to print an int array.

Parameters

a	: the array to be printed

5.21.2.7 sum_array()

```
double sum_array (
            std::vector< double > a)
```

Function to sum the elements of a double array.

Parameters

a	: the array to be summed

Returns

the sum of the elements of the array

5.22 src/matcher/io/io.cpp File Reference

This file contains the definitions of the functions used to write and read data to and from HDF5 files. The write_to↩ _hdf5_file function writes the data to an HDF5 file. The writeCloudData function writes the cloud data to the HDF5 file. The writeGroupDataChildren function writes the children data to the HDF5 file. The writeGroupDataParents function writes the parents data to the HDF5 file. The get_last_group function gets the last group in the group structure. The read_cloud_data_int function reads integer data from the HDF5 file. The read_cloud_data_double function reads double data from the HDF5 file.

```
#include "../../../include/matcher/io.h"
#include <hdf5.h>
#include <iostream>
#include <string>
#include <vector>
```

Functions

- void write_to_hdf5_file (CitySnaps &snapsnap, Params &params, int parent_num_clouds, int child_num_↩ clouds)

  This function writes the data to an HDF5 file.
- void writeCloudData (hid_t group_id, const MemberCloud &member, int child_flag)

  This function writes the cloud data to the HDF5 file.
- void writeGroupDataChildren (hid_t group_id, const std::string &subgroup_name, const MemberCloud &member)

  This function writes the children data to the HDF5 file.
- void writeGroupDataParents (hid_t group_id, const std::string &subgroup_name, const MemberCloud &member)

  This function writes the parents data to the HDF5 file.
- std::vector< int > read_cloud_data_int (Params &params, int snap_num, const std::string &field_to_read, const std::string &cloud_name)

  This function reads integer data from the HDF5 file.
- std::vector< double > read_cloud_data_double (Params &params, int snap_num, std::string &field_to_read, std::string &cloud_name)

  This function reads double data from the HDF5 file.
- hid_t get_last_group (Group_struct *subgroup_struct, hid_t group)

  This function gets the last group in the group structure.

5.22.1 Detailed Description

This file contains the definitions of the functions used to write and read data to and from HDF5 files. The write_to↩ _hdf5_file function writes the data to an HDF5 file. The writeCloudData function writes the cloud data to the HDF5 file. The writeGroupDataChildren function writes the children data to the HDF5 file. The writeGroupDataParents function writes the parents data to the HDF5 file. The get_last_group function gets the last group in the group structure. The read_cloud_data_int function reads integer data from the HDF5 file. The read_cloud_data_double function reads double data from the HDF5 file.

Author

Shivan Khullar

Date

June 2024

5.22.2 Function Documentation

5.22.2.1 get_last_group()

```
hid_t get_last_group (
            Group_struct * subgroup_struct,
            hid_t group)
```

This function gets the last group in the group structure.

Function to get the last group (traverse down the hdf5 file structure to the dataset)

Parameters

subgroup_struct	The Group_struct object containing the group structure.
group	The group ID of the group.

Returns

The last group in the group structure.

5.22.2.2 read_cloud_data_double()

```
std::vector< double > read_cloud_data_double (
            Params & params,
            int snap_num,
            std::string & field_to_read,
            std::string & cloud_name)
```

This function reads double data from the HDF5 file.

Parameters

params	The Params object containing the parameters of the program.
snap_num	The snapshot number.
field_to_read	The field to read from the HDF5 file.
cloud_name	The name of the cloud whose data to read.

Returns

The double data read from the HDF5 file.

5.22.2.3 read_cloud_data_int()

```
std::vector< int > read_cloud_data_int (
            Params & params,
            int snap_num,
            const std::string & field_to_read,
            const std::string & cloud_name)
```

This function reads integer data from the HDF5 file.

Parameters

params	The Params object containing the parameters of the program.
snap_num	The snapshot number.
field_to_read	The field to read from the HDF5 file.
cloud_name	The name of the cloud whose data to read.

Returns

The integer data read from the HDF5 file.

5.22.2.4 write_to_hdf5_file()

```
void write_to_hdf5_file (
            CitySnaps & snapsnap,
            Params & params,
            int parent_num_clouds,
            int child_num_clouds)
```

This function writes the data to an HDF5 file.

Parameters

snapsnap	The CitySnaps object containing the data to be written to the HDF5 file.
params	The Params object containing the parameters of the program.
parent_num_clouds	The number of parent clouds.
child_num_clouds	The number of child clouds.

5.22.2.5 writeCloudData()

```
void writeCloudData (
            hid_t group_id,
            const MemberCloud & member,
            int child_flag)
```

This function writes the cloud data to the HDF5 file.

Parameters

group_id	The group ID of the group to which the data is to be written.
member	The MemberCloud object containing the cloud data.
child_flag	The flag indicating whether the cloud is a child or a parent.

5.22.2.6 writeGroupDataChildren()

```
void writeGroupDataChildren (
            hid_t group_id,
            const std::string & subgroup_name,
            const MemberCloud & member)
```

This function writes the children data to the HDF5 file.

Parameters

group_id	The group ID of the group to which the data is to be written.
subgroup_name	The name of the subgroup to which the data is to be written.
member	The MemberCloud object containing the cloud data.

5.22.2.7 writeGroupDataParents()

```
void writeGroupDataParents (
            hid_t group_id,
            const std::string & subgroup_name,
            const MemberCloud & member)
```

This function writes the parents data to the HDF5 file.

Parameters

group_id	The group ID of the group to which the data is to be written.
subgroup_name	The name of the subgroup to which the data is to be written.
member	The MemberCloud object containing the cloud data.

5.23 src/matcher/main/matcher.cpp File Reference

This is the main driver of the matcher application. Its purpose is to match the particles among different cloud entities. It reads in the parameters from the xyz_params.txt file and assigns.

```
#include <iostream>
#include <string>
#include <hdf5.h>
#include <vector>
#include "../../../include/matcher/structs_and_classes.h"
#include "../../../include/matcher/matcher_functions.h"
#include "../../../include/matcher/read_params.h"
#include "../../../include/matcher/io.h"
#include "../../../include/matcher/utilities.h"
```

Functions

- int main (int argc, char **argv)

  Main driver. Calls the 'matcher' function after reading in the parameters. Reads in the parameter file and assigns parameters to the code variables.

5.23.1 Detailed Description

This is the main driver of the matcher application. Its purpose is to match the particles among different cloud entities. It reads in the parameters from the xyz_params.txt file and assigns.

Author

Shivan Khullar

Date

June 2024

5.23.2 Function Documentation

5.23.2.1 main()

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

Main driver. Calls the 'matcher' function after reading in the parameters. Reads in the parameter file and assigns parameters to the code variables.

Parameters

argc	Number of arguments
argv	Array of arguments

Returns

0 if successful, 1 if not

5.24 src/matcher/main/matcher_functions.cpp File Reference

Contains the definitions of the functions used in the matcher application. This includes the compare_particles, mass_frac, get_cloud_name, and load_clouds_to_group functions. The compare_particles function compares two particle ID lists and counts the number of matching IDs. The mass_frac function calculates the mass fraction of particles between two clouds. The get_cloud_name function returns the name of the cloud based on the cloud number. The load_clouds_to_group function loads the cloud data into the parent or child group. The matcher function is the main driver of the matcher application. It matches the particles among different cloud entities.

```
#include "../../../include/matcher/matcher_functions.h"
#include "../../../include/matcher/structs_and_classes.h"
#include "../../../include/matcher/read_params.h"
#include "../../../include/matcher/io.h"
#include "../../../include/matcher/utilities.h"
#include <hdf5.h>
#include <iostream>
#include <string>
#include <vector>
```

Functions

- int compare_particles (std::vector< double > a, std::vector< double > b)

  Compares two particle ID lists and counts the number of matching IDs.
- void mass_frac (std::vector< double > parent_pIDs, std::vector< double > child_pIDs, std::vector< double > parent_masses, std::vector< double > child_masses, int num_common_particles, double &childs_mass↩_frac_from_parent, double &parents_mass_frac_to_child)

  Calculates the mass fraction of particles between two clouds.
- std::string get_cloud_name (int i, Params &params)

  Returns the name of the cloud based on the cloud number.
- void load_clouds_to_group (int snap_num, int num_clouds_snap, Params &params, CitySnaps &snapsnap, std::string key)

  Loads the cloud data into the parent or child group.
- void matcher (Params &params)

  The main driver of the matcher application. Matches the particles among different cloud entities.

5.24.1 Detailed Description

Contains the definitions of the functions used in the matcher application. This includes the compare_particles, mass_frac, get_cloud_name, and load_clouds_to_group functions. The compare_particles function compares two particle ID lists and counts the number of matching IDs. The mass_frac function calculates the mass fraction of particles between two clouds. The get_cloud_name function returns the name of the cloud based on the cloud number. The load_clouds_to_group function loads the cloud data into the parent or child group. The matcher function is the main driver of the matcher application. It matches the particles among different cloud entities.

Author

Shivan Khullar

Date

June 2024

5.24.2 Function Documentation

5.24.2.1 compare_particles()

```
int compare_particles (
            std::vector< double > a,
            std::vector< double > b)
```

Compares two particle ID lists and counts the number of matching IDs.

Parameters

a	The first list of particle IDs.
b	The second list of particle IDs.

Returns

The number of matching particle IDs.

5.24.2.2 get_cloud_name()

```
std::string get_cloud_name (
            int i,
            Params & params)
```

Returns the name of the cloud based on the cloud number.

Function to get the cloud name by adding required number of zeros.

Parameters

i	The cloud number.
params	The parameters of the simulation.

Returns

The name of the cloud.

5.24.2.3 load_clouds_to_group()

```
void load_clouds_to_group (
            int snap_num,
            int num_clouds_snap,
            Params & params,
            CitySnaps & snapsnap,
            std::string key)
```

Loads the cloud data into the parent or child group.

Parameters

snap_num	The snapshot number.
num_clouds_snap	The number of clouds in the snapshot.
params	The parameters of the simulation.
snapsnap	The CitySnaps object.
key	The key to determine if the cloud is a parent or child.

5.24.2.4 mass_frac()

```
void mass_frac (
            std::vector< double > parent_pIDs,
            std::vector< double > child_pIDs,
            std::vector< double > parent_masses,
            std::vector< double > child_masses,
            int num_common_particles,
            double & childs_mass_frac_from_parent,
            double & parents_mass_frac_to_child)
```

Calculates the mass fraction of particles between two clouds.

Parameters

parent_pIDs	The list of particle IDs of the parent cloud.
child_pIDs	The list of particle IDs of the child cloud.
parent_masses	The list of masses of the parent cloud.
child_masses	The list of masses of the child cloud.
num_common_particles	The number of particles that are common between the two clouds.
childs_mass_frac_from_parent	The mass fraction of the child cloud, sourced from the parent cloud.
parents_mass_frac_to_child	The mass fraction of the parent cloud, donated to the child cloud.

5.24.2.5 matcher()

```
void matcher (
            Params & params)
```

The main driver of the matcher application. Matches the particles among different cloud entities.

Parameters

params	The parameters of the simulation.

Returns