fdcl-hdf5 Documentation

Release 0.0.1

Shankar Kulumani

Feb 04, 2020

INSTALLATION

CHAPTER

ONE

INSTALLATION

This library serves as an interface wrapper for using the HDF5 library with Eigen objects. As a result, all Eigen Matrix types are able to be written to HDF5 objects and subsequently read back. In addition, all HDF5 structures, such as files, groups, and datasets are compatible with Eigen data types.

CHAPTER

TWO

DEPENDENCIES

This library depends on the following:

1. Eigen - A C++ template library for linear algebra.
2. HDF5 - A file format to store and organize large amounts of data.
3. CMake - A cross-platform build tool.

CHAPTER

THREE

BUILD SCRIPTS

Several `bash` scripts are available which simplify the dependency installtion process. Usage of these scripts is not required if the dependencies are already installed or managed seperately.

CHAPTER

FOUR

BUILD AND INSTALLATION

CHAPTER

FIVE

BASIC USAGE

This file is a convinient header to include all of the HDF5 Eigen functionality into your program. You can utilize it by adding the following to your source code and ensuring that the library is available to your compiler

```
#include "hdf5.hpp"
```

Here is a simple program which shows how to write and read an Eigen matrix from an HDF5 data file.

```
#include "hdf5.hpp"
#include <Eigen/Dense>
#include <iostream>

void write_data() {
    Eigen::MatrixXd matrix(3, 3);
    matrix << 1, 2, 3, 4, 5, 6, 7, 8, 9;

    // open the file
    HDF5::File hf = HDF5::File("filename.hdf5", HDF5::File::Truncate);

    // write the data
    hf.write("dataset_name", matrix);

    std::cout << "Original Matrix: " << std::endl;
    std::cout << matrix << std::endl;
}

void read_data() {
    // open the file for reading
    HDF5::File hf = HDF5::File("filename.hdf5", HDF5::File::ReadOnly);

    Eigen::MatrixXd matrix;

    hf.read("dataset_name", matrix);

    std::cout << "Matrix read: " << std::endl;
    std::cout << matrix << std::endl;
}

int main() {
    write_data();
    read_data();
    return 0;
}
```

The `write_data()` function is used to write an `Eigen::MatrixXd` object to an `HDF5::File hf` instance.

Likewise, the `read_data()` function then opens the same file, in a read-only mode, and print the matrix to the screen.

CHAPTER

SIX

BASIC HDF5 TUTORIAL

Describe HDF5 File, Groups, Dataset ideas

Compare to file structure

Talk about attributes for data

Give some background on why it's better than other storage methods

CHAPTER

SEVEN

HDF5 FILE

This module provides functionality for dealing with `HDF5::File` objects.

class File

Public Functions

File(void)

virtual ~File(void)

File(**const** std::string &*file_name*, **const** int *open_flag* = *File*::ReadOnly)

const std::string **getName**(void) **const**

Group **group**(**const** std::string &*group_name*) **const**

DataSet **dataset**(**const** std::string &*dataset_name*) **const**

template<typename **Derived**>
DataSet **dataset**(**const** std::string &*name*, **const** Eigen::EigenBase<*Derived*> &*mat*) **const**

DataSet **open_dataset**(**const** std::string &*dataset_name*) **const**

template<typename **Derived**>
DataSet **read_dataset**(**const** std::string &*dataset_name*, **const** Eigen::DenseBase<*Derived*> &*mat*) **const**

template<typename **Derived**>
DataSet **write_dataset**(**const** std::string &*dataset_name*, **const** Eigen::EigenBase<*Derived*> &*mat*) **const**

template<typename **Derived**>
int **write**(**const** std::string &*dataset_name*, **const** Eigen::EigenBase<*Derived*> &*mat*)

template<typename **Derived**>
int **read**(**const** std::string &*dataset_name*, **const** Eigen::DenseBase<*Derived*> &*mat*)

Public Members

std::shared_ptr<H5::H5File> **file_ptr**
HDF5 file to save data

Public Static Attributes

const int **ReadOnly** = 0
Read only access

const int **ReadWrite** = 1
ReadWrite access

const int **Truncate** = 2
Overwrite a file if it exists or create a new one

const int **Excl** = 3
Only open if the file doesn't exist

const int **Create** = 4
Create a new file

CHAPTER

EIGHT

HDF5 GROUP

This module provides functionality for dealing with `HDF5::Group` objects.

class Group

Public Functions

Group(void)

virtual ~Group(void)

Group(**const** *File* *file, **const** std::string &group_name)

Group(**const** *Group* *group, **const** std::string &group_name)

Group **group**(**const** std::string &group_name) **const**

DataSet **dataset**(**const** std::string &name) **const**

template<typename **Derived**>
DataSet **dataset**(**const** std::string &name, **const** Eigen::EigenBase<*Derived*> &mat) **const**

template<typename **Derived**>
int **write**(**const** std::string &dataset_name, **const** Eigen::EigenBase<*Derived*> &mat)

template<typename **Derived**>
int **read**(**const** std::string &dataset_name, **const** Eigen::DenseBase<*Derived*> &mat)

Public Members

std::shared_ptr<H5::Group> **group_ptr**

CHAPTER

NINE

HDF5 DATASET

This module provides functionality for dealing with `HDF5::DataSet` objects.

class DataSet

Public Functions

DataSet(void)

virtual ~DataSet(void)

DataSet(**const** *File* *file, **const** std::string &dataset_name)

DataSet(**const** *Group* *group, **const** std::string &dataset_name)

template<typename **Derived**>
int **read**(**const** Eigen::DenseBase<*Derived*> &mat)

template<typename **Derived**>
int **write**(**const** Eigen::EigenBase<*Derived*> &mat)

template<typename **Derived**>
DataSet(**const** *File* *file, **const** std::string &dataset_name, **const** Eigen::EigenBase<*Derived*> &mat)

template<typename **Derived**>
DataSet(**const** *Group* *group, **const** std::string &dataset_name, **const** Eigen::EigenBase<*Derived*> &mat)

Public Members

std::shared_ptr<H5::DataSet> **dataset_ptr**

CHAPTER

TEN

HDF5 EIGEN WRAPPER

Many of the core HDF5 functionality is contained here. The more complicated details required to read and write Eigen objects happens here and should typically not be utilized directly

Functions

template<typename **Derived**>
void **load**(**const** H5::H5Location *&h5group*, **const** std::string *&name*, **const** Eigen::DenseBase<*Derived*> *&mat*)

This will read from an HDF5 file/group and save the data into an eigen matrix. One should have some idea about the eigen array before reading

Author Shankar Kulumani

Version 26 April 2018

Parameters

- `h5group`: H5 group or file to read from
- `name`: String name of the dataset
- `mat`: Eigen variable to save data

template<typename **Derived**>
void **save**(H5::H5Location *&h5group*, **const** std::string *&name*, **const** Eigen::EigenBase<*Derived*> *&mat*, **const** H5::DSetCreatPropList *&plist* = H5::DSetCreatPropList::DEFAULT)

This will write Eigen data to an HDF5 file or group

Author Shankar Kulumani

Version 26 April 2018

Parameters

- `h5group`: The H5 group or file to write to
- `name`: string name of the dataset
- `mat`: Eigen variable to save data
- `plist`: The dataset property list

template<typename **T**>

class DatatypeSpecialization

#include <hdf5_eigen.hpp> Convert a C++ datatype to the equivalent in HDF5 land.

All Eigen functionality to read/write to HDF5

Updated/corrected from the work of Jim Garrison https://github.com/garrison/eigen3-hdf5

Each dataset must define the correct type in the HDF5. This template class handles that by automatially determining the HDF5 type from a Eigen input.

Author Shankar Kulumani

Version 26 April 2018

Setup primiarly for Eigen types but needs to be extended.

Author Shankar Kulumani

Version 26 April 2018

template<>
struct DatatypeSpecialization<float>

#include <hdf5_eigen.hpp>

Public Static Functions

static const H5::DataType ***get**(void)

template<>
struct DatatypeSpecialization<double>

#include <hdf5_eigen.hpp>

Public Static Functions

static const H5::DataType ***get**(void)

template<>
struct DatatypeSpecialization<long double>

#include <hdf5_eigen.hpp>

Public Static Functions

static const H5::DataType ***get**(void)

template<>
struct DatatypeSpecialization<short>

#include <hdf5_eigen.hpp>

Public Static Functions

static const H5::DataType ***get**(void)

template<>
struct DatatypeSpecialization<unsigned short>
#include <hdf5_eigen.hpp>

Public Static Functions

static const H5::DataType ***get**(void)

template<>
struct DatatypeSpecialization<int>
#include <hdf5_eigen.hpp>

Public Static Functions

static const H5::DataType ***get**(void)

template<>
struct DatatypeSpecialization<unsigned int>
#include <hdf5_eigen.hpp>

Public Static Functions

static const H5::DataType ***get**(void)

template<>
struct DatatypeSpecialization<long>
#include <hdf5_eigen.hpp>

Public Static Functions

static const H5::DataType ***get**(void)

template<>
struct DatatypeSpecialization<unsigned long>
#include <hdf5_eigen.hpp>

Public Static Functions

static const H5::DataType ***get**(void)

template<>
struct DatatypeSpecialization<long long>
#include <hdf5_eigen.hpp>

Public Static Functions

static const H5::DataType ***get**(void)

template<>
struct DatatypeSpecialization<unsigned long long>
#include <hdf5_eigen.hpp>

Public Static Functions

static const H5::DataType ***get**(void)

template<>
struct DatatypeSpecialization<const char *>
#include <hdf5_eigen.hpp>

Public Static Functions

static const H5::DataType ***get**(void)

template<>
struct DatatypeSpecialization<char *>
#include <hdf5_eigen.hpp>

Public Static Functions

static const H5::DataType ***get**(void)

template<std::size_t **N**>
struct DatatypeSpecialization<const char[N]>
#include <hdf5_eigen.hpp>

Public Static Functions

static const H5::DataType ***get**(void)

template<std::size_t **N**>
struct DatatypeSpecialization<char[N]>
#include <hdf5_eigen.hpp>

Public Static Functions

static const H5::DataType ***get**(void)

namespace internal

Functions

template<typename **Derived**>
H5::DataSpace **create_dataspace**(**const** Eigen::EigenBase<*Derived*> &*mat*)
Create the dataspace required for the HDF5 dataset

Return H5::DataSpace Defines the space required to store mat

Author Shankar Kulumani

Version 26 April 2018

Parameters

- `mat`: Eigen matrix to write to the file

template<typename **Derived**>
bool **write_rowmat**(**const** Eigen::EigenBase<*Derived*> &*mat*, **const** H5::DataType ***const** *datatype*, H5::DataSet **dataset*, **const** H5::DataSpace **dspace*)

template<typename **Derived**>
bool **write_colmat**(**const** Eigen::EigenBase<*Derived*> &*mat*, **const** H5::DataType ***const** *datatype*, H5::DataSet **dataset*, **const** H5::DataSpace **dspace*)

template<typename **Scalar**>
void **read_data**(**const** H5::DataSet &*dataset*, *Scalar* **data*, **const** H5::DataType &*datatype*)

template<typename **Scalar**>
void **read_data**(**const** H5::Attribute &*dataset*, *Scalar* **data*, **const** H5::DataType &*datatype*)

template<typename **Derived**>
bool **read_colmat**(**const** Eigen::DenseBase<*Derived*> &*mat*, **const** H5::DataType ***const** *datatype*, **const** H5::Attribute &*dataset*)

template<typename **Derived**>
bool **read_colmat**(**const** Eigen::DenseBase<*Derived*> &*mat*, **const** H5::DataType ***const** *datatype*, **const** H5::DataSet &*dataset*)

template<typename **Derived**, typename **DataSet**>
void **_load**(**const** *DataSet* &*dataset*, **const** Eigen::DenseBase<*Derived*> &*mat*)

CHAPTER

ELEVEN

INDICES AND TABLES

- genindex
- modindex
- search

INDEX