h5tools-matlab Documentation

Sara Patterson - December 12, 2022

Contents

1 Getting Started **2**

2 Datasets **2**
- 2.1 Numeric . . . 3
- 2.2 Char . . . 3
- 2.3 String . . . 3
- 2.4 Cellstr . . . 4
- 2.5 Datetime . . . 4
- 2.6 Enumerated types . . . 4
- 2.7 Logical . . . 4
- 2.8 Table . . . 5
- 2.9 Struct . . . 5
- 2.10 Containers.Map . . . 5
- 2.11 Timetable . . . 5
- 2.12 Duration . . . 6
- 2.13 Miscellaneous data types . . . 6

3 Attributes **6**
- 3.1 Numeric . . . 7
- 3.2 Char . . . 7
- 3.3 String . . . 8
- 3.4 Logical . . . 8
- 3.5 Enumerated types . . . 8
- 3.6 Datetime . . . 8
- 3.7 Cellstr . . . 9
- 3.8 Duration . . . 9

4 Soft Links **9**

5 Groups **9**

6 Indexing **10**

7 Resources **10**

h5tools-matlab

The purpose of this documentation is to detail the implementation of reading and writing data with MATLAB and HDF5. The main functions are demonstrated here (and more succinctly in "docs/Examples.md"). Additional information on function inputs/outputs can be found in the documentation within within the code using `help` or `doc`.

1 Getting Started

Clone h5tools-matlab from Github (https://github.com/sarastokes/h5tools-matlab) or download from MATLAB's file exchange. Make sure to add the "src" folder and all subfolders to your MATLAB path as follows, replacing the ".." with the location where h5tools-matlab is saved.

```
addpath(genpath('../h5tools-matlab/src/'));
```

2 Datasets

The high-level functions for datasets are `h5tools.write` and `h5tools.read`. The implementation for these functions can be found in the "h5tools.datasets" package. The key difference between these functions and their corresponding MATLAB versions (`h5read` and `h5write`) are:

1. With MATLAB's library, you need to first create the dataset with `h5create` (which requires some understanding of HDF5 dataspaces), then write it with `h5write`. `h5tools.write` takes care of dataspaces and dataset creation behind the scenes.
2. Support for more MATLAB data types. Text (`char` and `string`), enumerated types, and compound datatypes (`table`, `timetable` and `struct/containers.Map` where each field has the same number of elements) are directly supported, in addition to the numeric types supported by MATLAB's high-level functions. A few tricks are used to support additional types, by converting to an HDF5-compatible format and adding attributes to the dataset that guide h5tools in reading the data back into MATLAB (`logical`, `cellstr`, `struct/containers.Map` with unequal elements in each field, `datetime`, `duration`).

The syntax is identical for each datatype:

```
h5tools.write(hdfName, pathName, dsetName, data);
out = h5tools.read(hdfName, pathName, dsetName);
```

The `hdfName` is the HDF5 file name, `pathName` is the path within the HDF5 file where the dataset will be written, `dsetName` is the dataset name and `data` is the data to be written.

```
% Write a dataset named 'MyDatatset' to' GroupOne'
h5tools.createFile('Test.h5');
h5tools.write('Test.h5', '/GroupOne', 'MyDatatset', magic(5));
out = h5tools.read('Test.h5', '/GroupOne', 'MyDataset');
```

Be sure to check the implementation for the data type you intend to write. In describing each datatype's implementation, the following information is provided where relevant:

- Preprocessing: Any changes to the data's MATLAB type that occur before writing to make the data compatible with an HDF5 datatype.
- Persistence: How the data is actually written to the HDF5 dataset.
- Postprocessing: Any processing of the data that occurs after reading into MATLAB to make the data compatible a specific MATLAB datatype.

- Implementation: The functions used to implement writing and reading of the dataset. All data passes through `h5tools.writeDatasetByType` for writing and `h5tools.readDatasetByType` for reading. Any additional functions implementing specialized processing will be detailed here.
- Limitations: Any caveats or limitations.

2.1 Numeric

MATLAB supports direct conversion of numeric datatypes to and from HDF5 files with `h5read` and `h5write`. The advantage of h5tools is a more user-friendly interface that automates dataset creation (`h5create`).

- Preprocessing: As with all other datatypes, the dataset creation and dataspace creation is handled rather than requiring the use of `h5create`. Writing is performed with `h5read`.
- Persistence: See the table below for the correspondence between MATLAB numeric types and HDF5 data types.

MATLAB type	HDF5 Class	HDF5 Type
double	H5T_FLOAT	H5T_IEEE_F64LE
single	H5T_FLOAT	H5T_IEEE_F32LE
uint8	H5T_INTEGER	H5T_STD_U8LE
int8	H5T_INTEGER	H5T_STD_I8LE
uint16	H5T_INTEGER	H5T_STD_U16LE
int16	H5T_INTEGER	H5T_STD_I16LE
uint32	H5T_INTEGER	H5T_STD_U32LE
int32	H5T_INTEGER	H5T_STD_I32LE
uint64	H5T_INTEGER	H5T_STD_U64LE
int64	H5T_INTEGER	H5T_STD_I64LE

- Postprocessing: N/A
- Implementation: `h5tools.datasets.makeMatrixDataset`, `h5read`
- Limitations: There are also several features that MATLAB's built-in functions already handle well and are not currently supported in h5tools, such as reading subsets of numeric datasets, using filters and working with remote locations.

2.2 Char

- Preprocessing: N/A
- Persistence: Written as class `H5T_STRING` with character type `H5T_C_S1` and set `H5T_CSET_ASCII`.
- Postprocessing: N/A
- Implementation: `h5tools.datasets.makeCharDataset`, `h5read`
- Limitations: N/A

2.3 String

- Preprocessing: N/A
- Persistence: written as class `H5T_STRING` with character type `H5T_C_S1` and set `H5T_CSET_UTF8`.
- Postprocessing: N/A
- Implementation: `h5tools.datasets.makeStringDataset`, `h5read`
- Limitations: N/A

2.4 Cellstr

While `cellstr` is not supported, it is comparable to a `string` array and can be written as such.

- Preprocessing: Conversion to `string` array.
- Persistence: Identical to `string`, but with 1 attribute: *`Class`* = `cellstr`.
- Postprocessing: Conversion from `string` back to `cellstr`
- Implementation: See `string`.
- Limitations: N/A

2.5 Datetime

While `datetime` is not supported (HDF5 has a time type but I have no looked into it yet), the data can be represented as a `string`, as long as the format is recorded.

- Preprocessing: The "format" property is extracted, then the data is converted to `string`.
- Persistence: See `string`. Two attributes are written to the dataset: *`Class`* = `datetime` and *`Format`* = `date format`.
- Postprocessing: Conversion to `datetime` using the optional Format argument.
- Implementation: `h5tools.datasets.makeDateDataset`
- Limitations: If providing an array of `datetime` values, they must have the same format. Also, the conversion with `datestr` rounds the seconds, so if millisecond precision is important, this will be problematic. A future alternative would be to encode day, month, year, hours, seconds etc. as attributes instead to avoid rounding.

2.6 Enumerated types

HDF5 offers enumerated type, which is an integer datatype restricted to a set of named values. For example, an enumerated type may contain three members (e.g. "Red", "Green", "Blue") and their corresponding values (0, 1, 2).

- Preprocessing: The index of each enumerated type value within the full list of enumerated types (obtained with `metaclass`) is converted to `uint32`. The class name (including any namespaces) will be appended to each enumeration name (e.g. "RED" will be written as "MyClassName.Red").
- Persistence: written as `H5T_ENUM` with the values of each member in the enumeration as `H5T_STD_U32LE`.
- Postprocessing: A string array of the enumeration members is read from the dataset through the datatype library (`H5T`), ordered by their enumeration value. If the associated class is not found on MATLAB's path, the data will be read back in as a string array using the format `x.y` where `x` is the class name in the attributes and `y` is the enumeration name. If the class is found on MATLAB's path, the string will be evaluated (`eval`). If the enumerated type was not written with h5tools and has no class name attribute, the string array will just contain the enumeration names.
- Implementation: `h5tools.datasets.makeEnumTypeDataset`, `h5tools.datasets.readEnumTypeDataset`.
- Limitations:

2.7 Logical

HDF5 has no boolean datatype so, following other successful HDF5 libraries like h5py, this is written as an enumerated type.

- Preprocessing: Conversion to `uint8`.
- Persistence: written as `H5T_ENUM` with two members (0=FALSE, 1=TRUE). The enumeration values are written as `H5T_STD_I8LE`.

- Postprocessing: If an enumerated type dataset is read in and has two enumerated types matching those above, the values it will be treated as **logical**. The enumerated values will be converted from **uint8** to **logical**.
- Implementation: `h5tools.datasets.makeLogicalDataset`, `h5tools.datasets.readEnumTypeDataset`.
- Limitations: N/A

2.8 Table

HDF5 supports writing tables as compound datatypes, for which there is no MATLAB support.

- Preprocessing:
- Persistence: written as H5T_COMPOUND with two attributes: *Class* = table, *ColumnClasses* = [class1, class2,...] of data in each column
- Postprocessing:
- Implementation: `h5tools.datasets.makeCompoundDataset`, `h5tools.datasets.readCompoundDataset`
- Limitations: Currently only numeric datatypes, string and char are supported within the table.

2.9 Struct

Both **struct** and **containers.Map** mimic the natural layout of an HDF5 file (ie.e they are a group containing datasets) so there are naturally difficulties writing them to a single dataset.

- Preprocessing: The number of elements in each field will be calculated to determine the strategy used to write the information to the HDF5 file.
- Persistence: If the number of elements in each field is the same, H5T_COMPOUND is used, as described above, but with the identifying attribute *Class* = struct. Otherwise, each field will be written as an attribute of a placeholder **string** dataset containing the original MATLAB class ("struct").
- Postprocessing: See **table**. If the number of elements was not equal and the dataset was mapped to attributes instead, the attributes and their values will be assigned to the fields of a **struct**.
- Implementation: `h5tools.datasets.makeStructDataset`, `h5tools.datasets.makeMapDataset`
- Limitations: As described above, the number of elements in each field must be the same. Multi-level structs (e.g. a **struct** where one or more of the fields are **struct**. The limitations of **table** also apply.

2.10 Containers.Map

Containers.Map is treated similarly to **struct**.

- Preprocessing: Conversion to **struct** with `map2struct`.
- Persistence: See **struct**. Only difference is, if the dataset is mapped to attributes, the placeholder text dataset will contain "containers.Map".
- Postprocessing: If necessary, conversion with `struct2map()`.
- Implementation: `h5tools.datasets.makeStructDataset`, `h5tools.datasets.makeMapDataset`
- Limitations: See **struct** and **table**.

2.11 Timetable

- Preprocessing: The time column is converted to **double** with `seconds()`.
- Persistence: Written as H5T_COMPOUND with two attributes: *Class* = timetable, *ColumnClasses* = [class1, class2,...] of data in each column. See **Table 1** and **string** and **char** for information on how individual columns are written.

- Postprocessing: The first column is converted back to **duration** with `seconds`.
- Implementation: See **table**.
- Limitations: Same as **table**. In addition, a better way might exist for converting the **duration** column than always using `seconds`.

2.12 Duration

- Preprocessing: All durations are converted to seconds, then to **double**
- Persistence: Written as `H5T_IEEE_F64LE` with one attribute: *Class* = duration.
- Postprocessing: Converted back to **duration** with `seconds()`.
- Implementation: `h5write`
- Limitations: N/A

2.13 Miscellaneous data types

Several specialized MATLAB datatypes from the Image Processing Toolbox are also included to demonstrate how the tricks above could be applied to write any other specialized datatype to an HDF5 file as well. See `h5tools.datasets.writeDatasetByType` and `h5tools.datasets.readDatasetByType` for implementation. See CONTRIBUTING.md for information about adding new types.

3 Attributes

The high-level functions for attributes are `h5tools.writeatt` and `h5tools.readatt`. Many of the classes supported by h5tools for datasets rely on attributes to ensure accurate conversion when reading in the data. Because attributes cannot have attributes, the additional datatypes available here are less extensive and some differ in their implementation. The advantages of `h5tools.writeatt` and `h5tools.readatt` are:

1. Expanded support for MATLAB datatypes including **numeric**, **char**, **string**, **logical**, **datetime** and enumerated types. Several other datatypes will write to HDF5 but will not be read back in as the same datatype (**cellstr** and **duration**). If you try to write an attribute without write support, you'll get an error. If you write one without read support, a warning will be sent to the command line specifying the path within the HDF5 file, attribute name and datatype, so that you are aware.
2. The ability to work with one or more attributes at a time, or to read all the attributes of a dataset/group without knowing their names.

The options for creating attributes with `h5tools.writeatt` are demonstrated below.

```
% Write a single attribute:
h5tools.writeatt('Test.h5', '/', 'A', 1);

% Write multiple attributes:
h5tools.writeatt('Test.h5', '/', 'A', 1, 'B', 2);

% Write all the fields of a struct as attributes
attStruct = struct('A', 1, 'B', 2);
h5tools.writeatt('Test.h5', '/', attStruct);

% Write all the key/value pairs in a containers.Map:
attMap = containers.Map('A', 1, 'B', 2);
h5tools.writeatt('Test.h5', '/', attMap);

% Write a struct or containers.Map followed by additional attributes:
h5tools.writeatt('Test.h5', '/', attMap, 'C', 3);
h5tools.writeatt('Test.h5', '/', attStruct, 'C', 3);
```

The options for reading attributes with `h5tools.readatt` are demonstrated below.

```
% 1. Read a single attribute, return the value
out = readatt('File.h5', '/GroupOne', 'Attr1')

% 2. Read two attributes, return the values as two outputs
[out1, out2] = readatt('File.h5', '/GroupOne', 'Attr1', 'Attr2')

% 3. Read two attributes, return the values in a containers.Map:
out = readatt('File.h5', '/GroupOne', 'Attr1', 'Attr2')

% 4. Read all attributes, return the values in a containers.Map
out = readatt('File.h5', '/GroupOne', 'all')
```

The implementation for numeric datatypes, `char`, `string`, `logical` and enumerated types is largely identical to that described above for datasets. All attributes pass through `h5tools.attributes.writeAttributeByType` and `h5tools.attributes.readAttributeByType`. Any additional functions are noted in the "Implementation" points below.

3.1 Numeric

MATLAB supports direct conversion of numeric datatypes to and from HDF5 files with `h5readatt` and `h5writeatt`. The advantage of h5tools is a more user-friendly interface that enables working with multiple attributes as once, as discussed above.

- Preprocessing: As with all other datatypes, the dataset creation and dataspace creation is handled rather than requiring the use of `h5create`. Writing is performed with `h5read`.
- Persistence: See the table below for the correspondence between MATLAB numeric types and HDF5 data types.

MATLAB type	HDF5 Class	HDF5 Type
double	H5T_FLOAT	H5T_IEEE_F64LE
single	H5T_FLOAT	H5T_IEEE_F32LE
uint8	H5T_INTEGER	H5T_STD_U8LE
int8	H5T_INTEGER	H5T_STD_I8LE
uint16	H5T_INTEGER	H5T_STD_U16LE
int16	H5T_INTEGER	H5T_STD_I16LE
uint32	H5T_INTEGER	H5T_STD_U32LE
int32	H5T_INTEGER	H5T_STD_I32LE
uint64	H5T_INTEGER	H5T_STD_U64LE
int64	H5T_INTEGER	H5T_STD_I64LE

- Postprocessing: N/A
- Implementation: `h5writeatt`, `h5readatt`
- Limitations: For some reason, row vectors are being returned as columns (even though columns vectors are also returned as columns)

3.2 Char

Basically the same as `char` datasets

- Preprocessing: N/A
- Persistence: Written as class `H5T_STRING` with character type `H5T_C_S1` and set `H5T_CSET_ASCII`.
- Postprocessing: N/A
- Implementation: `h5writeatt`, `h5readatt`
- Limitations:

3.3 String

Basically the same as `string` datasets

- Preprocessing: N/A
- Persistence: Written as class `H5T_STRING` with character type `H5T_C_S1` and set `H5T_CSET_UTF8`.
- Postprocessing: N/A
- Implementation: `h5writeatt`, `h5readatt`
- Limitations:

3.4 Logical

HDF5 has no boolean datatype so, following other successful HDF5 libraries like h5py, this is written as an enumerated type.

- Preprocessing: Conversion to `int8`.
- Persistence: written as `H5T_ENUM` with two members (0=FALSE, 1=TRUE), where the member values are `H5T_STD_I8LE`.
- Postprocessing: If an enumerated type dataset is read in and has two enumerated types matching those above, the values it will be treated as `logical`. The enumerated values will be converted from `uint8` to `logical`.
- Implementation: `h5tools.datasets.makeLogicalAttribute`, `h5tools.datasets.readEnumTypeAttribute`.
- Limitations: N/A

3.5 Enumerated types

HDF5 offers enumerated type, which is an integer datatype restricted to a set of named values. For example, an enumerated type may contain three members (e.g. "Red", "Green", "Blue") and their corresponding values (0, 1, 2).

- Preprocessing: The index of each enumerated type value within the full list of enumerated types (obtained with `metaclass`) is converted to `uint32`. The class name (including any namespaces) will be appended to each enumeration name (e.g. "RED" will be written as "MyClassName.Red").
- Persistence: written as `H5T_ENUM` with the values of each member in the enumeration as `H5T_STD_U32LE`.
- Postprocessing: A string array of the enumeration members is read from the dataset through the datatype library (`H5T`), ordered by their enumeration value. If the associated class is not found on MATLAB's path, the data will be read back in as a string array using the format `x.y` where `x` is the class name in the attributes and `y` is the enumeration name. If the class is found on MATLAB's path, the string will be evaluated (`eval`). If the enumerated type was not written with h5tools and has no class name attribute, the string array will just contain the enumeration names.
- Implementation: `h5tools.datasets.makeEnumTypeAttribute`, `h5tools.datasets.readEnumTypeAttribute`.
- Limitations:

3.6 Datetime

Because attributes cannot be used to specify `datetime` information, as done for datasets, the process for `datetime` attributes is slightly different.

- Preprocessing: Conversion to `string` and the format property is appended to each string
- Persistence: `H5T_STRING` with character type `H5T_C_S1` and set `H5T_CSET_UTF8`.
- Postprocessing: Conversion back to `datetime`.

- Implementation: `h5tools.attributes.makeDatetimeAttribute`
- Limitations: The seconds are rounded when converting to **`string`**.

3.7 Cellstr

While **`cellstr`** is not supported, it is comparable to a **`string`** array and can be written as such.

- Preprocessing: Conversion to **`string`** array.
- Persistence: `H5T_STRING` with character type `H5T_C_S1` and set `H5T_CSET_UTF8`.
- Postprocessing: N/A
- Implementation: See **`string`**.
- Limitations: The data cannot be read back in as **`cellstr`** and will be returned as a **`string`** array instead.

3.8 Duration

- Preprocessing: All durations are converted to seconds, then to **`double`**
- Persistence: Written as `H5T_IEEE_F64LE`.
- Postprocessing: N/A
- Implementation:
- Limitations: The data cannot be read back in as **`duration`** and will be returned as **`double`** instead.

4 Soft Links

Soft links are comparable to "shortcuts" within a file system. Currently, h5tools supports writing a dataset that is a soft link to another dataset or group within the HDF5 file (`h5tools.writelink`). Arrays of soft links and soft link attributes are not supported.

Soft links are created as demonstrated below. **`hdfFile`** is the HDF5 file name, **`linkLocation`** is the path for the group where the link will be written, **`linkName`** is the name of the link and **`targetLocation`** is the HDF5 path of the group/dataset that will be linked to.

```
% SYNTAX:
% h5tools.writelink(hdfFile, linkLocation, linkName, targetLocation);

% Add a soft link within group '/G1' that links to group '/G2'.
h5tools.writelink('File.h5', '/G1', 'Link1', '/G2');
```

5 Groups

MATLAB's high-level functions create groups en route to creating datasets but does not offer direct control of group creation. The `h5tools.createGroup` function fills this gap by providing a high-level option for creating one or more groups, as demonstrated below.

The first input (**`hdfFile`**) is the HDF5 file name and the second input (**`hdfPath`**) is the HDF path where the groups will be written. Additional inputs (**`varargin`**) are group names. Groups do not need to be within **`hdfPath`**, but their group names must be specified relative to this location within the file.

```
% SYNTAX
% h5tools.createGroup(hdfFile, hdfPath, varargin)

% Create one group within the root group "/"
h5tools.createGroup('Test.h5', '/', 'GroupOne);

% Create three groups within "/GroupOne"
h5tools.createGroup('Test.h5', '/GroupOne', 'GroupOne1A', 'Group1B', 'Group1C');
```

```
% Create two groups, one in the root group "/" and one in "/GroupOne"
h5tools.createGroup('Test.h5', '/', 'GroupTwo', '/GroupOne/Group1D');
```

6 Indexing

Several functions are provided for indexing the contents of an HDF5 file. More details about each can be found in the documentation within each function.

1. `h5tools.collectDatasets` - Return the full path names of all datasets in an HDF5 file.
2. `h5tools.collectGroups` - Return the full path names of all groups in an HDF5 file.
3. `h5tools.collectSoftlinks` - Return the full path names of all soft links in an HDF5 file.
4. `h5tools.getAttributeNames` - Return the names of all attributes of a specific group or dataset.

7 Resources

- MATLAB: See their documentation for the high-level functions as well as the low-level library.
- HDF5 API Reference: The official HDF5 documentation.
- NeurodataWithoutBorders: nwb-matlab has HDF5 support for a number of data types. Although their HDF5 functions are specific to NWB, it's still a great resource for learning about HDF5 I/O. They have support for object and region references, which is something I would like to add in the future.
- h5py: an excellent Python library for HDF5 – if your project isn't tied to MATLAB, consider using this instead. For example, idea to make `logical` an enumerated type was borrowed from h5py.