Export to '.tmat.hdf5' format with R

baptiste

This document describes a few R utility functions to export T-matrices in the standardised HDF5 format. For illustration, we start by producing a dummy dataset.

Mockup input data

The `rhdf5` package has support for a variety of R objects, including lists which are automatically written as grouped objects in HDF5.

```
# possibly multiple wavelengths
wavelength <- seq(400, 800, by=50)
Nl <- length(wavelength)

# dummy 30x30 matrix values for each wavelength
tmatrix <- array(1:900 + 1i*(1:900), c(30,30,Nl))

modes <- list('l' = 1:30,'m' = 1:30,
              'polarization' =  rep(c('electric','magnetic'),15))

# dummy 'analytical zeros' for e.g. EBCM methods
zeros <- expand.grid(p=seq(1,30,by=2), pp = seq(1,30,by=2))

materials <- list('embedding' = list('relative_permeability' = 1.0,
                                     'relative_permittivity' = 1.33^2),
                  'Au' = list('relative_permeability' = 1.0,
                              'relative_permittivity' = -11.4+1.181i))

geometry <- list('shape' = 'spheroid','radiusxy' = 20.0, 'radiusz' = 40.0)

# details about computation, including full script
script <-  glue::glue_collapse(readLines('test-dummy.R'), sep = "\n")
computation <- list('method' ='EBCM',
```

```
                      'software' = 'SMARTIES',
                      'version' = '1.1',
                      'Ntheta' = 40,
                      'accuracy' = 1e-10,
                      'analytical_zeros' = zeros,
                      'script' = script)

uuid <- uuid::UUIDgenerate()
hdf5version <- glue::glue_collapse(H5get_libversion(),".")
```

Saving to HDF5

The `rhdf5` package provides support for reading/writing HDF5 files into/from R objects. Until my recent request, complex arrays were not supported, but this is now implemented in the dev version of the package. The `glue` and `purrr` packages are only used for convenience, and can easily be replaced with base functions.

```
library(rhdf5) # note: requires dev version to support complex arrays
# cf https://support.bioconductor.org/p/9156305/#9156408
# install.packages("remotes")
# remotes::install_github("grimbough/rhdf5@devel")
# may need 'crypto' library from openSSL to compile
# e.g. via brew install openssl on macos

library(uuid) # uuid
library(glue) # string interpolation
library(purrr) # mapping functions
```

```
## create file
f <- 'ar.tmat.h5'
unlink(f)
h5createFile(f)
h5closeAll()
```

We can then write the different objects defined above using `hwrite`. Note the important `native = TRUE` parameter for the T-matrix data: R stores arrays column-wise, while HDF5 uses row-major ordering. To avoid confusion between different programming languages, we suggest sticking with the native HDF5 convention (`native = TRUE` ensures that the array is written transposed).

```
## write file
h5write(tmatrix, file=f, name="/tmatrix", native=TRUE) # store rowwise
# Using native = TRUE increases HDF5 file portability between programming languages
h5write(modes, file=f, name='/modes')
h5write(wavelength, file=f, name="/vacuum_wavelength")
h5write(embedding, file=f, name="/embedding")
h5write(materials, file=f, name="/materials")
h5write(geometry, file=f, name="/geometry")
h5write(computation, file=f, name='/computation')
h5write(uuid, file=f, name='/uuid')
```

Attributes are written in a separate step, and a convenience wrapper function `write_attributes(object, names, attributes, type)` is used to simplify the interface.

```
## open file
fid <- H5Fopen(f)
## root attributes
write_attributes("/",
                 names = list("name",
                              "created_with",
                              "storage_format_version",
                              "description",
                              'keywords'),
                 attributes = list("Au prolate spheroid in water",
                                   "rhdf5",
                                   hdf5version,
                    "Computation using SMARTIES, a robust EBCM for spheroids", 'gold, sphero
                 type = "root")

## object attributes
write_attributes("tmatrix",
                 names=c("units","created_with"),
                 attributes = c("nm","rhdf5"),
                 type = "dataset")

write_attributes("vacuum_wavelength",
                 names=c("unit"),
                 attributes = c("nm"),
                 type = "dataset")

write_attributes("uuid",
                 names=c("version"),
```

```
                  attributes = c("4"),
                  type = "dataset")

write_attributes("geometry",
                  names=c("name"),
                  attributes = c("prolate spheroid"),
                  type = "group")
## close file
H5Fclose(fid)
```