Export to '.tmat.hdf5' format with Matlab

baptiste

This document describes a few Matlab utility functions to export T-matrices in the standardised HDF5 format. For illustration, we start by producing a dummy dataset. The SMARTIES implementation of EBCM for spheroids uses these tools internally for its HDF5 export.

Mockup input data

The `easyh5` library takes care of most of the details for us, when objects are stored in Matlab structures. There are a couple of caveats, illustrated below, such as `polarization` being handled separately, and attributes being added at a later stage.

```
% possibly multiple wavelengths
wavelength  = (400:50:800)';
Nl = length(wavelength);

% dummy 30x30 matrix values for each wavelength
tdata = reshape((1:900) + 1i*(1:900), [30,30]);
tmatrix = repmat(tdata, [1,1,Nl]);

% modes, but note that polarization is handled separately
modes = struct('l', 1:30, 'm', 1:30);
polarization = repmat(["electric","magnetic"], 1, 15);

% dummy 'analytical zeros' for e.g. EBCM methods
[zerosp, zerospp] = ndgrid(1:2:30, 1:2:30);
zeros  = struct('p', zerosp, 'pp', zerospp);

% materials
embedding = struct('relative_permeability', 1.0, ...
                   'relative_permittivity', 1.33^2);
particle = struct('relative_permeability', 1.0, ...
                  'relative_permittivity', -11.4+1.181i);
```

```
materialname = 'Au';
materials = struct('embedding', embedding, materialname, particle);

% geometry
geometry = struct('shape', 'spheroid', 'radiusxy', 20.0, 'radiusz', 40.0);

% details about computation, including full script
computation = struct('method','EBCM',...
    'software','SMARTIES',...
    'version','1.1',...
    'Ntheta', 40, ...
    'accuracy','1e-10', ...
    'analytical_zeros', zeros, ...
    'script', fileread('test_dummy.m')); % embed full script as string

% combined (almost all) information into one struct
a = struct('tmatrix', tmatrix, ...
    'vacuum_wavelength', wavelength, ...
    'embedding', embedding,...
    'materials', materials, ...
    'geometry', geometry, ...
    'modes', modes, ...
    'computation', computation, ...
    'uuid', char(matlab.lang.internal.uuid()));

[maj,min,rel] = H5.get_libversion();
hdf5version = sprintf('%d.%d.%d',maj,min,rel);
```

Saving to HDF5

```
addpath(genpath('../easyh5/'));
```

`saveh5` does most of the work, but we have to write `polarization` separately as arrays of strings within structs seem to trip `easyh5`. I did not find how to link the embedding medium from materials to the top level, so it is simply duplicated.

```
% save to file
f = 'am.tmat.h5';
saveh5(a, f, 'ComplexFormat', {'r','i'}, 'rootname', '', ...
       'Compression', 'deflate'); % compression is optional
```

```
% write polarization separately
h5create(f,'/modes/polarization', size(polarization), 'Datatype', 'string')
h5write(f,'/modes/polarization', polarization)
```

Attributes are written in a separate step.

```
% write root attributes
h5writeatt(f, '/', 'name', 'Au prolate spheroid in water');
h5writeatt(f, '/', 'created_with', 'Matlab easyh5');
h5writeatt(f, '/','keywords', 'gold, spheroid, ebcm');
h5writeatt(f, '/', 'storage_format_version', hdf5version);
h5writeatt(f, '/','description', ...
    'Computation using SMARTIES, a robust EBCM for spheroids');

% attributes of specific objects
h5writeatt(f, '/vacuum_wavelength', 'unit', 'nm');
h5writeatt(f, '/uuid', 'version', '4');
h5writeatt(f, '/geometry', 'name', 'prolate spheroid');
```