List of changes to the EasySpin Code in order to run it on Octave

Nicolás I. Neuman

June 1st 2021

This document, as well as the code in this repository, is provided without any warranty whatsoever. This is a working draft that will be updated as more functionality is added and larger parts of Easyspin are tested.

On running '`easyspin`', the first warning is on `verLessThan`, called from `chkmlver`, from `easyspininfo`, which checks the Matlab version. These checks have to be commented everywhere on the code.

First important change:

Comment

```
error(chkmlver);
```

everywhere.

The following error is in:

```
easyspincompile at line 34 column 4
```

This error occurs because the function

```
mex.getCompilerConfigurations('C','Installed');
```

which checks that there is a compiler installed, is not understood. The function `mex` cannot be indexed with a dot in Octave.

Solution: Comment lines 34-39. But we have to make sure that Octave has access to the proper C-compiler, so that the mex-files will be compiled.

After commenting this line, compilation of 9 files is peformed. In my installation, there is one file that fails:

(4/9) mdhmm_lbfgsb_wrapper.c

The full trace is the following:

```
>> easyspin
```

```
EasySpin not yet compiled for this platform (0/9 files). Attempting to
compile.


EasySpin compilation

  directory:                C:\Users\User\Documents\Intec\Investigacion\EPR
Simulation\EasySpin-1jun2021\easyspin\private

  version: 6.2.0

  mex extension: mex, 32-bit

  compiling 9 c-files...

  (1/9) chili_lm.c                  complete

  (2/9) cubicsolve.c                complete

  (3/9) lisum1i.c                   complete

  (4/9)  mdhmm_lbfgsb_wrapper.c        mdhmm_lbfgsb_wrapper.c:111:1:  error:
unknown type name 'mxLogical'; did you mean 'logical'?

  111 | mxLogical isInt( const mxArray *pm ) {

      | ^~~~~~~~~

      | logical

mdhmm_lbfgsb_wrapper.c:111:24: error: unknown type name 'mxArray'

  111 | mxLogical isInt( const mxArray *pm ) {

      |                        ^~~~~~~

mdhmm_lbfgsb_wrapper.c: In function 'isInt':

mdhmm_lbfgsb_wrapper.c:126:20:  warning:  implicit  declaration  of  function
'mxIsInt16' [-Wimplicit-function-declaration]

  126 |             return mxIsInt16(pm);

      |                    ^~~~~~~~~

mdhmm_lbfgsb_wrapper.c:128:20:  warning:  implicit  declaration  of  function
'mxIsInt32' [-Wimplicit-function-declaration]

  128 |             return mxIsInt32(pm);

      |                    ^~~~~~~~~

mdhmm_lbfgsb_wrapper.c:130:20:  warning:  implicit  declaration  of  function
'mxIsInt64' [-Wimplicit-function-declaration]

  130 |             return mxIsInt64(pm);

      |                    ^~~~~~~~~

mdhmm_lbfgsb_wrapper.c:132:13:  warning:  implicit  declaration  of  function
'mexErrMsgTxt' [-Wimplicit-function-declaration]

  132 |            mexErrMsgTxt("You have a weird computer that I don't know
how to support");

      |             ^~~~~~~~~~~
```

```
mdhmm_lbfgsb_wrapper.c:133:20: error: 'false' undeclared (first use in this
function); did you mean 'fclose'?
  133 |             return false;
      |                    ^~~~~
      |                    fclose
mdhmm_lbfgsb_wrapper.c:133:20: note: each undeclared identifier is reported
only once for each function it appears in
mdhmm_lbfgsb_wrapper.c: At top level:
mdhmm_lbfgsb_wrapper.c:139:29: error: unknown type name 'mxArray'
  139 | void mexFunction( int nlhs, mxArray *plhs[], int nrhs, const
mxArray*prhs[] )   {
      |                             ^~~~~~~
mdhmm_lbfgsb_wrapper.c:139:62: error: unknown type name 'mxArray'
  139 | void mexFunction( int nlhs, mxArray *plhs[], int nrhs, const
mxArray*prhs[] )   {
      |                                                              ^~~~~~~
warning: mkoctfile: building exited with failure status
  failed
mex: building exited with failure status
  (5/9) multimatmult_.c             complete
  (6/9) multinucstick.c             complete
  (7/9) projecttriangles.c          complete
  (8/9) projectzones.c              complete
  (9/9) sf_peaks.c                  complete


EasySpin compilation unsuccessful.


=========================================================================
  Release:          $ReleaseID$ ($ReleaseDate$)
  Expiry date:      $ExpiryDate$
  Folder:                   C:\Users\User\Documents\Intec\Investigacion\EPR
Simulation\EasySpin-1jun2021\easyspin
  MATLAB version:   6.2.0
  Platform:         Microsoft Windows [Version 10.0.19042.985]
  mex-files:        mex, 0.111111/
  System date:      01-Jun-2021 10:12:07
  Temp dir:         C:\Users\User\AppData\Local\Temp\
```

==========

Solution: For now, ignore this problem, as this file is necessary to implement the L-BFGS-B algorithm, that probably is used within the `esfit` function.

Updated Solution.

In `mdhmm_lbfgsb_wrapper.c`, uncomment

#include "mex.h"

Then I will run an example, which calls the function `pepper`.

```
chkmlver at line 15 column 1
pepper at line 61 column 1
```

Solution:

Comment

```
error(chkmlver);
```

everywhere.

After this, `pepper` gives warnings:

```
warning: implicit conversion from matrix to sq_string
pepper at line 113 column 1
pepper at line 158 column 33
```

Solution:

Change

```
error(err) -> error(strvcat(err));
```

everywhere

Within `pepper`, the `validatespinsys` function is called, which runs `easyspincompile`. Presumably as this compilation has failed for one file, it is not validated and tries to compile again.

So in the `validatespinsys` file we should comment lines 36-42.

Other warnings and errors in pepper

```
warning: implicit conversion from matrix to sq_string
warning: called from
    pepper at line 528 column 3
    pepper at line 158 column 33
    HSCo_pepper at line 34 column 12

warning: implicit conversion from matrix to sq_string
warning: called from
    parseoption at line 25 column 3
    pepper at line 561 column 24
    pepper at line 158 column 33
    HSCo_pepper at line 34 column 12

error: verLessThan: package "matlab" is not installed
error: called from
    verLessThan at line 72 column 5
    chkmlver at line 15 column 1
    resfields at line 45 column 1
    pepper at line 643 column 38
    pepper at line 158 column 33
    HSCo_pepper at line 34 column 12     resfields at line 45
column 1
```

Solution:
Everywhere in `pepper, parseoption`:
Change
`Error(err)`
by
`error(strvcat(err)); %Matlab->Octave`

In resfields (and to save time, also on resfields_perturb), comment `error(chkmlver)` and also fix `error(err);`

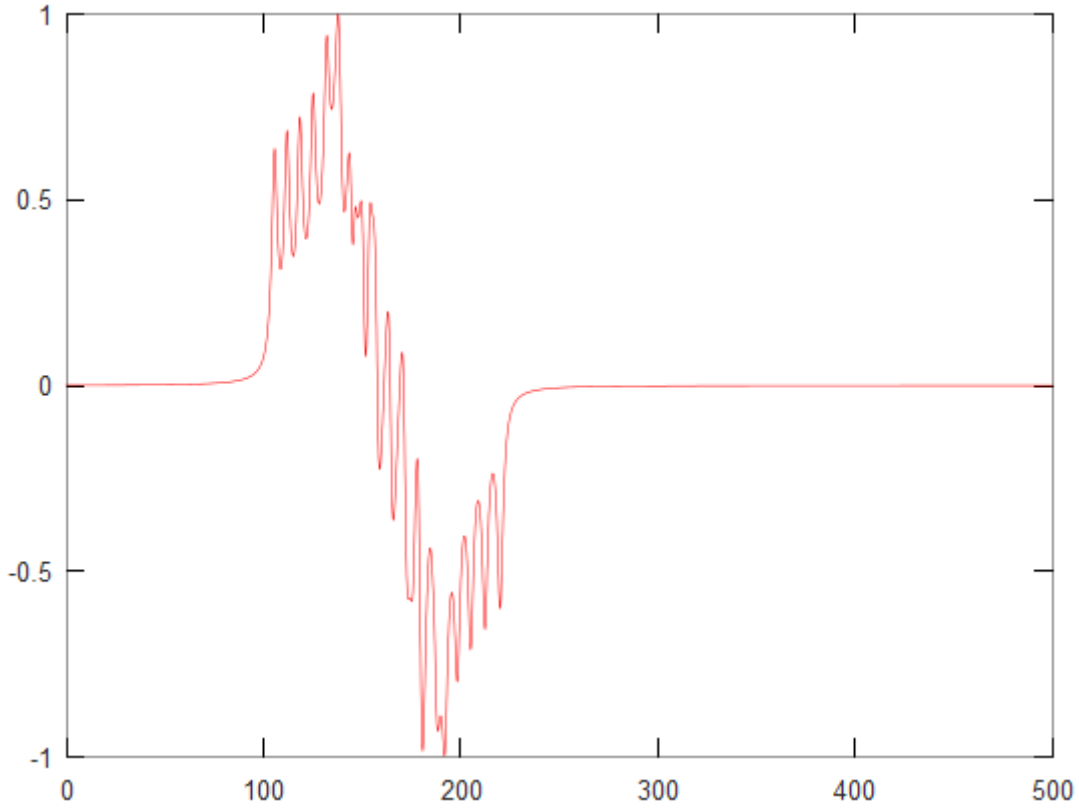
After fixing all these things, the following example passed:

`HSCo_pepper_onlySym.m`

```
clear all, close all

Bmin = 0; %in mT
Bmax = 500;
Freq = 9.456; %Frecuencia en Hz
```

```
g1 = [5.197 0 0;0 4.134 0;0 0 3.4];

Sys.S = 0.5;
Sys.g = g1;
Sys.Nucs = '59Co';
Sys.A = [500 0 0;0 400 0;0 0 300]; %MHz
Sys.lwpp = [0 2]; %mT

Vary.g = [0.2 0 0;0 0.2 0;0 0 0.2];
Vary.A = [50 0 0;0 50 0;0 0 50];

Exp.Range = [Bmin Bmax];
Exp.nPoints = 8192;
Exp.mwFreq = Freq;

[B,SimSpc] = pepper(Sys,Exp);

plot(B,SimSpc/max(SimSpc),'-r');
```

Produces the following simulation.

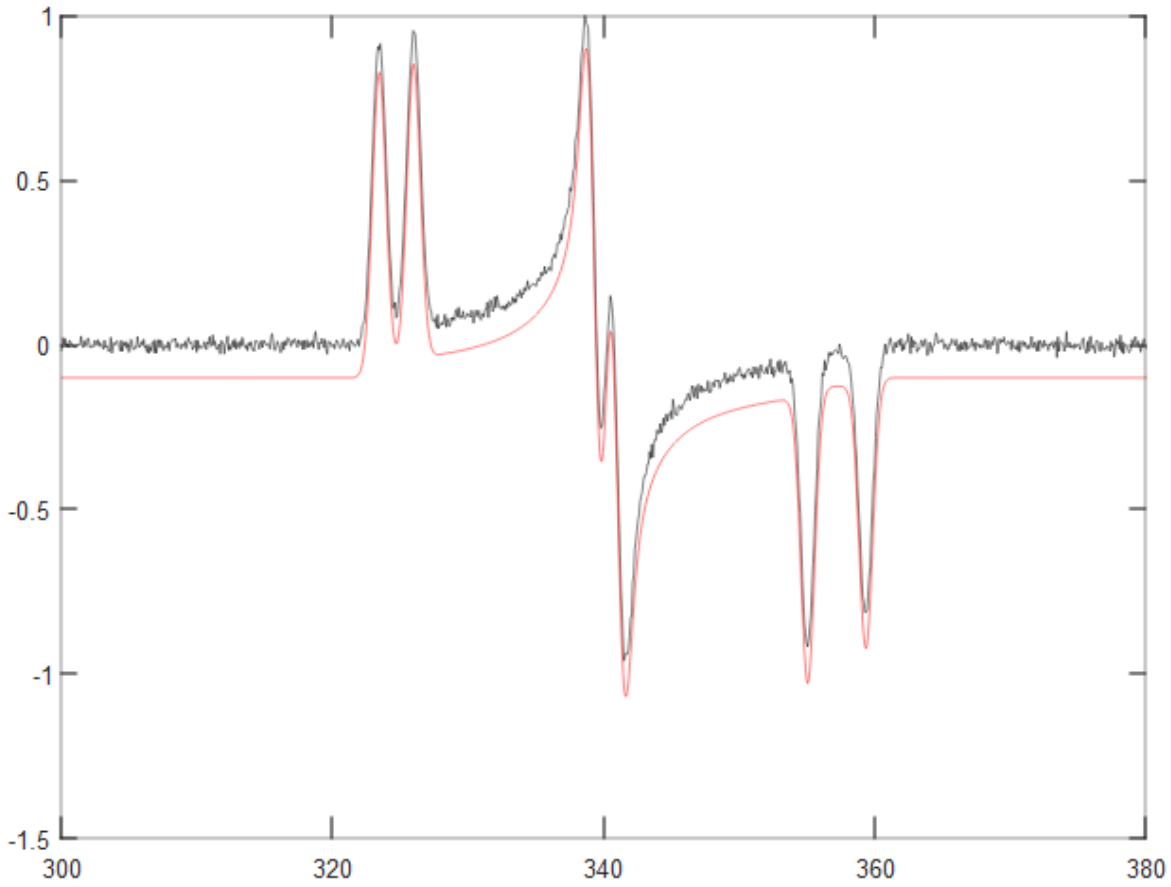


Another example which uses least-squares fiting (`esfit`) is the following (a small modification of `basicfit.m`)

```
% Basic example of spectral fitting using EasySpin
```

```
%======================================================================
clear

% Since we don't have any experimental data available, let's create some mock
% data by simulating a spectrum and adding some noise. If you use this example
% as a basis for your fittings, replace this code by code that loads your
% experimental data.

Sys.g = [2 2.1 2.2];
Sys.Nucs = '1H';
Sys.A = [120 50 78];
Sys.lwpp = 1;
Exp.mwFreq = 10;
Exp.Range = [300 380];
[B,spc] = pepper(Sys,Exp);
spc = addnoise(spc,150,'n');

% Now we set up the least-squares fitting. First comes a starting set of
% parameters (which we obtain by copying the spin system from the simulation
% and changing a few values).
Sys0 = Sys;
Sys0.A = [100 50 78];
Sys0.g(1) = 1.98;

% Next, we specify which parameter we want to be fitted and by how much the
% fitting algorithm can vary it approximately.
Vary.g = [0.1 0 0];
Vary.A = [30 0 0];

% We also can provide options for the simulation function.
SimOpt.Method = 'perturb';

% Finally, we specify the fitting algorithm and what should be fitted.
FitOpt.Method = 'simplex int'; % simplex algorithm, integrals of spectra

[bestsys,bestspc] = esfit(@pepper,spc,Sys0,Vary,Exp,SimOpt,FitOpt);

plot(B,spc/max(spc),'-k',B,bestspc/max(bestspc)-0.1,'-r');

% If the fitting algorithm doesn't find the correct minimum, you can change
% the algorithm, target function, and starting point in the UI. For example,
% run Monte Carlo for a bit, save the best fit, and then use that as the
% starting point with Nelder/Mead to close in on the correct fit.
```

This produces the following output.




```
>> bestsys
bestsys =

  scalar structure containing the fields:

    g =

       1.9999    2.1000    2.2000

    Nucs = 1H
    A =

       119.924     50.000     78.000

    lwpp = 1
    weight = 1
```