Matlab interfacing fortran

Interfacing with Fortran

To use fortran functions from matlab there are a few options:

- I. the loadlibrary method
- II. the mexfile method
- III. the swig method

I. The loadlibrary method

Prerequisites:
.so/.dll -> the dynamic library with the prebuild fortran code
.h -> the header file describing the functions and arguments in c notation

> ⚠ Fortran doesn't use text based header files but binary mod files. These are not recognized by matlab.

To generate the h files look in the dll or shared object what the naming convention generated. Type the names in the .h file. If you used fortran to generate the dll the names might and function arguments might be mangled. Also the function arguments might change. See the following example of a simple header file.

endecdll.h

```
#ifdef __cplusplus
extern "C" {
#endif
  /* entry point declarations here */
  __declspec( dllexport ) long fortomatint  (long& , long&);
                        # long fortomatint_ (long&, long&); \-> for g77
  __declspec( dllexport ) float fortomatreal(float& , float&);

#ifdef __cplusplus
} /* end extern "C" */
#endif
```

This header file is applicable for the following fortran code when compiled to dll. Note that you have to add the compiler directives to your fortran code for every fortran function you want to be visible in the dll.

endecdll.for

```
INTEGER*4 FUNCTION fortomat(val1,val2)
!DEC$ ATTRIBUTES DLLEXPORT,ALIAS : "fortomat" :: fortomat
INTEGER*4 val1,val2
FORTOMAT=val1*val2
END FUNCTION

real*4 FUNCTION fortomatreal(val1,val2)
!DEC$ ATTRIBUTES DLLEXPORT,ALIAS : "fortomatreal" :: fortomatreal
real*4 val1,val2
fortomatreal=val1*val2
END FUNCTION
```

> ✓ Use nm (linux) or depends (windows) to look inside a dynamic library.

To use the library within matlab you have to tell matlab about the library and the header file. Matlab should check if the naming convention used in the header file is consistent with the naming in the dll/so. See the following example:

example.m

```
if libisloaded('lib')~=1
if ispc
   loadlibrary endecdll.dll endecdll.h alias lib
else % lunix
   loadlibrary libendecdll.so endecdll.h alias lib % \-> for linux
end;
%libfunctions lib libfunctions lib \-full
%calllib('lib','fortomatint',5,5)
%calllib('lib','fortomatreal',5.4,3.7)
if libisloaded('lib')==1
unloadlibrary lib;
end;
```

You can make matlab wrapper that loads the library, calls the function, and unloads the libray again. With this matlab wrapper, the matlab doe snot even need to be aware that he is calling a ddll:

fortomat.m

```
function out = fortomat(var1,var2)
%FORTOMAT
%
% fortomat(val1,val2)
% where var* can be integer or float

   OPT.libname      = 'ekufbqegjhbgoebgo';
   OPT.dllname      = 'endecdll.dll';;
   OPT.hdrname      = 'endecdll.h'
   OPT.debug        = 1;

   %% Load library
   %% ------------
   if libisloaded(OPT.libaliasname)~=1
   loadlibrary(OPT.dllname,OPT.hdrname,'alias',OPT.libname)
   end;

   %% Inquire library
   %% ------------
   if OPT.debug
   libfunctions(OPT.libaliasname,'-full')
   end

   %% Call library function
   %% ------------
   if isa(var1,'integer') && isa(var1,'integer')
   out = calllib(OPT.libname,'fortomat',var1,var2);
   elseif isa(var1,'float') && isa(var1,'float')
   out = calllib(OPT.libname,'fortomat',var1,var2);
   else
   error(['No matching datatype in ',])
   end

   %% Unload library
   %% ------------
   if libisloaded(OPT.libname)==1
    unloadlibrary(OPT.libname);
   end;
%% EOF
```

This method is described in detail at the mathworks site

II. The mexfile method

This method uses the mex files (matlab specific dll's) to communicate between matlab and fortran. You make fortran (or c) aware of the internals of matlab. You wrap your method with the matlab function calling method (lhs, rhs, etc...) .
Matlab Fortran
See a simple example.

example.for

```
!
! CHECK FOR PROPER NUMBER OF ARGUMENTS
!
IF (NRHS .NE. 2) THEN
CALL MEXERRMSGTXT('YPRIME requires two input arguments')
ELSEIF (NLHS .GT. 1) THEN
CALL MEXERRMSGTXT('YPRIME requires one output argument')
ENDIF

!
! CHECK THE DIMENSIONS OF Y.  IT CAN BE 4 X 1 OR 1 X 4.
!
M = MXGETM(PRHS(2))
N = MXGETN(PRHS(2))
!
IF ((MAX(M,N) .NE. 4) .OR. (MIN(M,N) .NE. 1)) THEN
CALL MEXERRMSGTXT('YPRIME requires that Y be a 4 x 1 vector')
ENDIF
```

III. Using SWIG.

SWIG is an open source program to use native c/c++/fortran routines from dynamic and modern languages. This is used in tcl/python/c#/R and many other languages. There has been some work to extend this method to matlab. This however doesnt seem to be completely implemented.

SWIG & Matlab
SWIG->octave
SWIG -> Matlab

Authors: Fedor Baart, Jan Kramer & Gerben de Boer (transferred from McTools website).