IPOPT Installation on Windows 10

IPOPT is widely used to solve nonlinear programming problems [1,2]. However, it is notoriously difficult to install, especially on Windows. The original guide is available in https://coin-or.github.io/Ipopt/INSTALL.html; it is outdated for installation on Windows 10. The instructions below are an attempt to make the installation process easier for Windows users and those unfamiliar with Linux commands and C++. This guide was written during the installation of Ipopt version 3.13.3 on Windows 10 Home version 1909.

Prerequisites

Before installing IPOPT, each user must request for a COIN-HSL license via http://www.hsl.rl.ac.uk/ipopt/, to obtain the necessary linear solvers. If you are an academic, you can request for an academic license [click on “Source” under the “Academic License” section to fill out your details and request]. The user will then receive an email within 1 – 2 days, which contains a link to download a Zip file containing the license and related documentation. The user should download this file and extract the folder containing “coinhsl” in the name, for e.g. “coinhsl-2015.06.03”. The user can store this file temporarily in a desired location in the computer, until it is required (as the instructions state).

Guide

N.B. The “user” in the folder paths shown in instructions below (such as in *C:\msys64\home\user\Ipopt\* may change on each computer. For example, if the user’s name on the computer is John, then the folder path may be *C:\msys64\home\john\Ipopt\*.

1) Download MSYS2 on https://www.msys2.org/ and install it on the computer. MSYS2 is software that enables source code designed for Unix-like operating systems to be compiled and run natively on Windows.

2) Once installed, Run **MSYS2 MinGW 64-bit** as administrator (search MSYS2 on the Taskbar >> open file location >> right-click on **MSYS2 MinGW 64-bit** >> Run as administrator).

3) Install the following dependencies by typing the following into the MSYS2 command interface:
```
pacman -S binutils diffutils git grep make patch pkg-config
```

4) Now the installation of various packages can commence. You can copy and paste the following code given for each of these packages (only right click >> paste works on MYSYS2).

5) To install GCC try:
```
pacman -S mingw-w64-x86_64-gcc
```

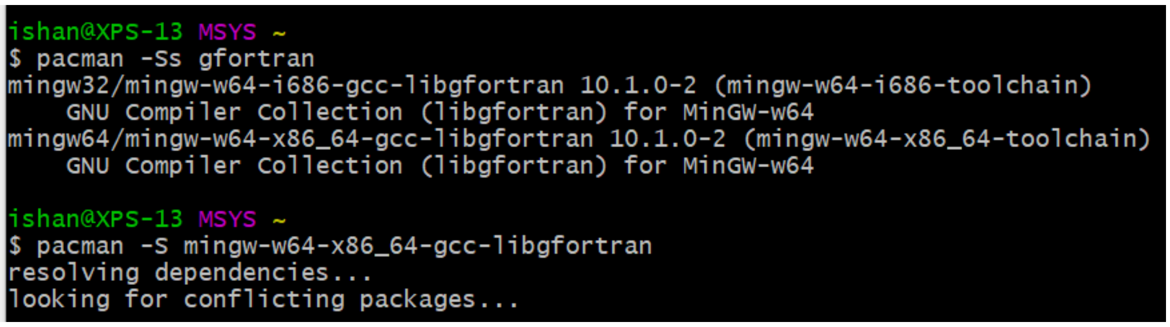
   N.B. If any of this code returns an error, you can search the package using the following code:
```
pacman -Ss packagename
```

   For example, to search for gfortran using the above statement and identify the name (this can be also applied to other packages when an error is returned)

*Written by Ishanki De Mel & Michael Short**
**m.short@surrey.ac.uk*

For example, if the module returns the following for gfortran when

`pacman -Ss gfortran` is used:



```
ishan@XPS-13 MSYS ~
$ pacman -Ss gfortran
mingw32/mingw-w64-i686-gcc-libgfortran 10.1.0-2 (mingw-w64-i686-toolchain)
    GNU Compiler Collection (libgfortran) for MinGW-w64
mingw64/mingw-w64-x86_64-gcc-libgfortran 10.1.0-2 (mingw-w64-x86_64-toolchain)
    GNU Compiler Collection (libgfortran) for MinGW-w64

ishan@XPS-13 MSYS ~
$ pacman -S mingw-w64-x86_64-gcc-libgfortran
resolving dependencies...
looking for conflicting packages...
```

Use:

`pacman –S mingw-w64-x86_64-gcc-libgfortran`

to install it (as shown in the image above, always use the one with mingw-w64-x86_64… flag.).

6) Install gfortran as shown in the example above.

7) To install metis try:
`pacman -S mingw-w64-x86_64-metis`

8) To install lapack try:
`pacman -S mingw-w64-x86_64-lapack`

9) Open your file explorer and search for the following link:
C:\msys64\mingw64\bin
This is where all the extensions for the packages are. This folder should be added to the Path/Environmental Variables on your PC (to do this, copy the link, right click This PC>> Properties>> Advanced System Settings>> Advanced>> Environment Variables>> Path>> Edit>> New.
More instructions are available here: https://www.correlatedsolutions.com/support/index.php?/Knowledgebase/Article/View/85/1/running-python-scripts-from-anywhere-under-windows)

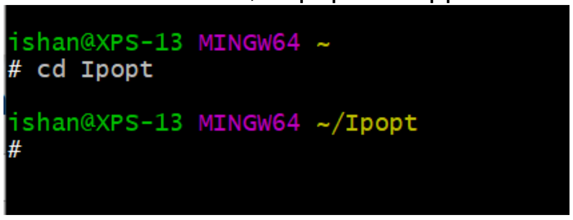
N.B. It is useful to keep this window open so you can check whether the extensions are installed and whether your folders are being installed in the right place.

10) The following must also be added to the Path/Environmental Variables as shown above:
C:\msys64\mingw64\include
C:\msys64\mingw64\lib

11) Download IPOPT through GITHUB using:
`git clone https://github.com/coin-or/Ipopt.git`

12) Test: transfer to the Ipopt folder using
`cd Ipopt`

If this is successful, ~/Ipopt will appear on MSYS2 as shown below:



```
ishan@XPS-13 MINGW64 ~
# cd Ipopt

ishan@XPS-13 MINGW64 ~/Ipopt
#
```

N.B. If you now go on *C:\msys64\home\user* via File Explorer you will notice that there is an "Ipopt" folder there. It is useful to keep an eye on the contents in and around this folder.

13) Come out of the Ipopt folder using

```
cd ..
```

14) Make sure you get the Third Party-ASL using

```
git clone https://github.com/coin-or-tools/ThirdParty-ASL.git
cd ThirdParty-ASL
./get.ASL
./configure
Make
Make install
```

N.B. You can check for the new "ThirdParty-ASL" folder installed in *C:\msys64\home\user.*
Also, if you get any errors at this point, it is very likely that you haven't installed one of the packages such as GCC, Gfortran, etc. properly. Make sure to check the extensions in *C:\msys64\mingw64\bin* and re-install if these extensions are not found.

15) Now go to the unzipped "coinhsl" folder you received from COIN-HSL (for e.g. "coinhsl-2015.06.03"). Make sure to rename it as simply "coinhsl".

16) On MSYS2 use the following code to create a ThirdParty-HSL folder in *C:\msys64\home\user*

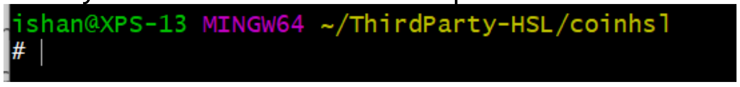
```
git clone https://github.com/coin-or-tools/ThirdParty-HSL.git
cd ThirdParty-HSL
```

17) Through File Explorer, copy-paste the coinhsl folder into the ThirdParty-HSL folder found in
C:\msys64\home\user

18) On MSYS2, now go into the folder you just copy pasted using:

```
cd coinhsl
```

19) Once you have confirmed that the path looks like this:

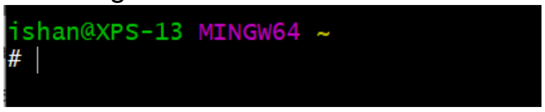


```
ishan@XPS-13 MINGW64 ~/ThirdParty-HSL/coinhsl
# |
```

You can now configure the HSL using the code below:

```
./configure
make
make install
```

20) Come out of the coinhsl and ThirdParty-HSL folders using
`cd ..`
Twice.
Or
`cd ../..`
The image below shows what it would look like after you've done this:



```
ishan@XPS-13 MINGW64 ~
# |
```

21) Now install MUMPs using the following code

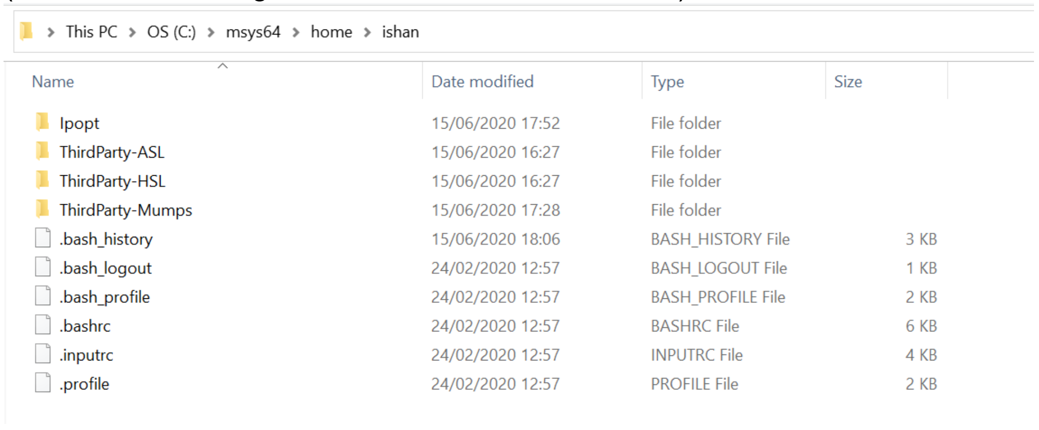
```
git clone https://github.com/coin-or-tools/ThirdParty-Mumps.git
cd ThirdParty-Mumps
./get.Mumps
./configure
```

(N.B. due to a MUMPS bug in the current update, `./configure` alone does not currently work. It should be accompanied by the flag `ADD_FCFLAGS=-fallow-argument-mismatch`, otherwise an error is returned. Other possible flags that might work: `./configure ADD_FCFLAGS=-std=legacy`; see https://github.com/coin-or-tools/ThirdParty-Mumps/issues/4 for more information. As it is an error in the current update, this issue may be resolved soon.)

22) Complete the MUMPS installation using

```
make
make install
```

23) Check if the HSL, ASL and MUMPS folders are in the correct location, i.e. in *C:\msys64\home\user*
(as seen in the image below where "user" is "ishan".)



Name	Date modified	Type	Size
Ipopt	15/06/2020 17:52	File folder	
ThirdParty-ASL	15/06/2020 16:27	File folder	
ThirdParty-HSL	15/06/2020 16:27	File folder	
ThirdParty-Mumps	15/06/2020 17:28	File folder	
.bash_history	15/06/2020 18:06	BASH_HISTORY File	3 KB
.bash_logout	24/02/2020 12:57	BASH_LOGOUT File	1 KB
.bash_profile	24/02/2020 12:57	BASH_PROFILE File	2 KB
.bashrc	24/02/2020 12:57	BASHRC File	6 KB
.inputrc	24/02/2020 12:57	INPUTRC File	4 KB
.profile	24/02/2020 12:57	PROFILE File	2 KB

If they are not, cut and paste the folders in the correct location (this should work, but if this does not, you will have to re-install them in the correct location)

24) On MYSYS2, come out of the ThirdParty-Mumps folder using
`cd ..`

25) Now go into the Ipopt folder using
`cd Ipopt`

*Written by Ishanki De Mel & Michael Short**
**m.short@surrey.ac.uk*

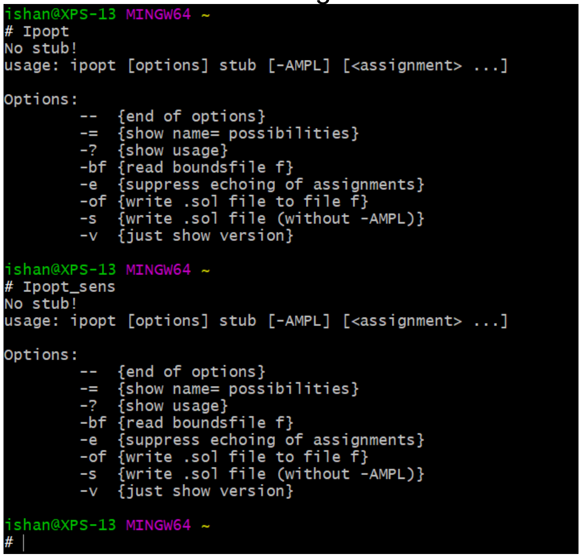
26) Now complete the Ipopt configuration using the code below:

```
./configure
Make
Make test
Make install
```

27) Once the installation is complete, check whether you have installed IPOPT correctly use the following commands on MSYS2:

```
Ipopt
Ipopt_sens
```

You will see the following if it is successful:



```
ishan@XPS-13 MINGW64 ~
# Ipopt
No stub!
usage: ipopt [options] stub [-AMPL] [<assignment> ...]

Options:
        --  {end of options}
        -=  {show name= possibilities}
        -?  {show usage}
        -bf {read boundsfile f}
        -e  {suppress echoing of assignments}
        -of {write .sol file to file f}
        -s  {write .sol file (without -AMPL)}
        -v  {just show version}

ishan@XPS-13 MINGW64 ~
# Ipopt_sens
No stub!
usage: ipopt [options] stub [-AMPL] [<assignment> ...]

Options:
        --  {end of options}
        -=  {show name= possibilities}
        -?  {show usage}
        -bf {read boundsfile f}
        -e  {suppress echoing of assignments}
        -of {write .sol file to file f}
        -s  {write .sol file (without -AMPL)}
        -v  {just show version}

ishan@XPS-13 MINGW64 ~
# |
```

28) Check in the Command Prompt as well (search for Command Prompt on the Taskbar):

```
Ipopt
Ipopt_sens
```

29) You now need to verify that IPOPT has detected the linear solvers on coinhsl. You can either run a programme on Python/Pyomo to do so and/or check whether the IPOPT configuration log has detected coinhsl. To access this log, go to
C:\msys64\home\user\Ipopt\
on your File Explorer and click on the "config" text document.
If you search for "HSL" within the txt you will see something like

```
#define IPOPT_HAS_HSL 1
```

Or

```
configure:23596: checking for package HSL
configure:23809: result: yes
```

If the HSL is not detected, then go back to instruction (16) and repeat till the end (skipping the MUMPS installation, which should have been successful the first time), ensuring that the HSL is in the correct location.

Congratulations! You have now installed IPOPT, one of the world's finest nonlinear programming solvers.

*Written by Ishanki De Mel & Michael Short**
**m.short@surrey.ac.uk*

References:

[1] Wächter A. An Interior Point Algorithm for Large-Scale Nonlinear Optimization with Applications in Process Engineering. PhD Thesis, Carnegie Mellon University, 2002.

[2] Wächter A, Biegler LT. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Math Program 2006;106:25–57. https://doi.org/10.1007/s10107-004-0559-y.

*Written by Ishanki De Mel & Michael Short**
**m.short@surrey.ac.uk*