f2py import Fortran code in Python

10 June, 2019

Importing Fortran code in Python just like any other Python module is very straightforward, using F2py.

Prereqs

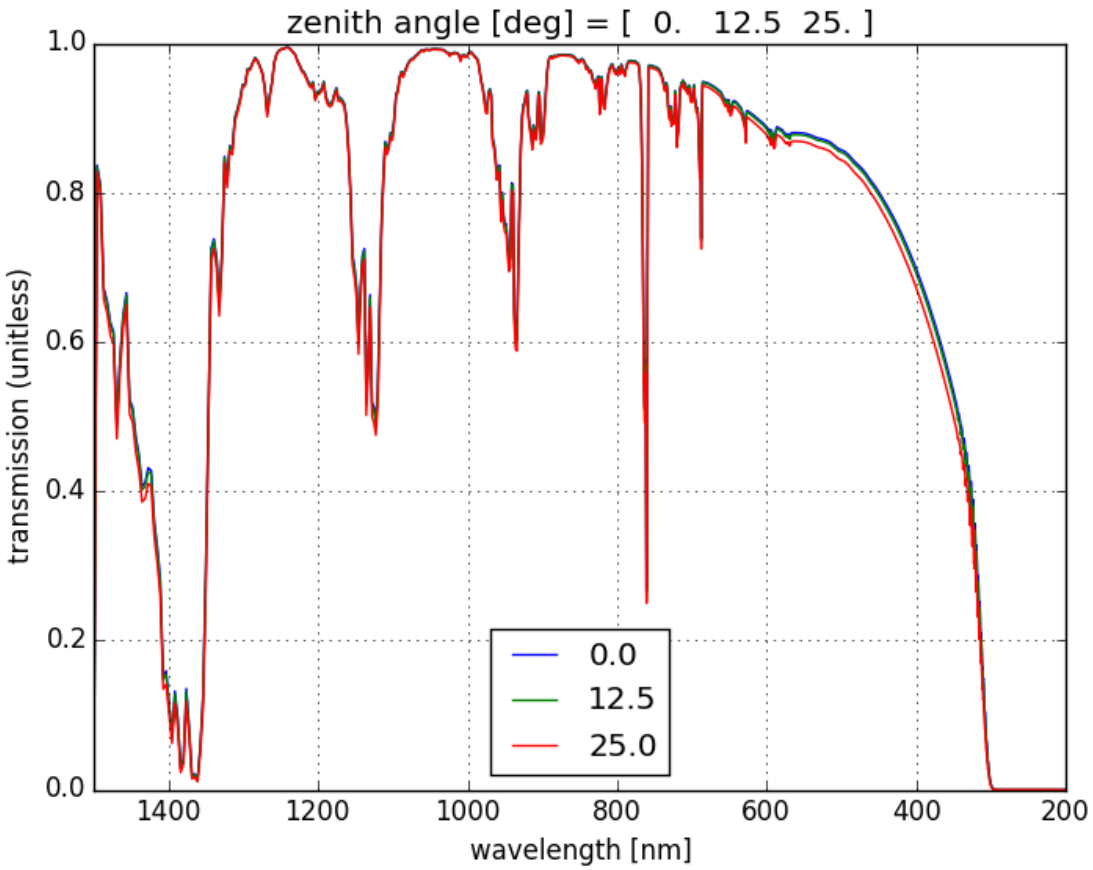
On any operating system, a Fortran compiler and Numpy are required to use F2py. If you don't already have a Fortran compiler, we suggest GNU Gfortran.

- MacOS / Linux: using Homebrew `brew install gcc`
- Linux / Windows Subsystem for Linux: `apt install gfortran`
- Windows: use MSYS2 `pacman -S mingw64/mingw-w64-x86_64-gcc-fortran`

Test/fix

Try the lowtran7 code. Following the instructions there, you should get

1. A `lowtran7.cp37-win_amd64.pyd` (on Windows) or `lowtran7*.so` (on Mac/Linux) file
2. Running `python DemoLowtran.py` creates a plot of atmospheric loss



Lowtran output

f2py does not allow inline comments for COMMON blocks

`f2py` does not allow inline comments for COMMON blocks for Fortran 77 `.f` code. This is because `f2py` works more strictly to Fortran specifications than most modern compilers.

Inline comments are *not* Fortran 77 standard, and will make `f2py` throw an error.

To fix this problem, just make the inline comment a full-line command with `!` in column 1.

Fortran90 `.f90` files won't throw an f2py error due to inline comments on a line with a COMMON block: goodcomment.f90.

This will manifest itself two different ways, depending on whether you have `implicit none` or not:

COMMON inline comment error WITH implicit none

Example in badcomment_implicit.f:

> var2fixfortran: No typespec for argument “x ! bad for fortran77”. getctype: No C-type found in “{}”, assuming void. KeyError: ‘void’

Solution: Make inline comment a full-line comment with `!` in column 1.

COMMON inline comment error WITHOUT implicit none

Example in badcomment.f

> error: expected ‘;’, ‘,’ or ‘)’ before ‘!’ token

Solution: Make inline comment a full-line comment with `!` in column 1: goodcomment.f.

Windows troubleshooting

Another solution is to use Windows Subsystem for Linux with Anaconda Python. However, with the techniques below, I’ve always gotten f2py to work on Windows.

Tell Python to use MinGW by creating file ~/pydistutils.cfg containing:

```
[build]
compiler=mingw32
```

Do this from Powershell by copy/paste this line:

```
echo "[build]`ncompiler=mingw32" | Out-File -Encoding ASCII ~/pydis
```

> ‘f2py’ is not recognized as an internal or external command, operable program or batch file.

Ensure Numpy is installed

```
conda install numpy
```

(Windows) create a file `<Anaconda Python install directory>/Scripts/f2py.bat` with content

```
python %~dp0/f2py.py %*
```

> Error: No module named 'numpy.distutils._msvccompiler' in numpy.distutils; trying from distutils

is fixed by: create file ~/pydistutils.cfg as above

> error LNK2001: unresolved external symbol _gfortran_st_write error LNK2001: unresolved external symbol _gfortran_st_read

and similar errors are typically from not having told f2py to use gfortran by: create file ~/pydistutils.cfg as above

---

If you don't want to create ~/pydistutils.cfg as recommended above, you can instead do for each F2py package you install:

```
python setup.py build_ext --inplace --compiler=mingw32
```

If you have problems using f2py or other development work on Windows, consider Windows Subsystem for Linux, which runs at full performance within a terminal window on Windows.

Notes

Reference

Related

simple F2py Fortran examples

python

fortran **f2py** **mingw**

GitHub Twitter