MoogPy: A Primer

MoogPy is a python library created with f2py that wraps the MoogStokes FORTRAN 77 code. With the library, python programs can directly access the results of the spectral synthesis without worrying about generating/curating/maintaining line lists, parameter files, and output files in the proper format and/or location.

MoogStokes functionality currently available in MoogPy:

- Use the Synth driver
  - `'synth'` – pure, unadulterated Moog
  - Given a list of absorption lines and a model atmosphere, Moog calculates the emergent Stokes I spectrum.
- Use the SynStokes driver
  - `'synstok'` – vintage Moog plus Polarized Radiative Transfer (PRT).
  - Given a list of absorption lines and a model atmosphere, MoogStokes calculates the emergent Stokes I, Q, U, and V spectra. As the emergent spectrum changes depending on the viewing angle and magnetic field orientation, it is necessary to divide up the stellar disk into segments, calculate the emergent spectrum from each segment, and later combine the emergent spectra into a composite spectrum, applying appropriate weights and wavelength shifts. The current version of the code assumes a uniform and radial magnetic field. Two disk integration strategies are currently available:
  - Beachball (only Stokes I and V) assumes solid-body rotation and divides the stellar disk into vertical stripes (in a vague analogy to a beachball).
    - Beachball uses Jeff Valenti's diskint algorithm
  - Diskoball (all Stokes vectors) divides the entire stellar surface into a configurable number of rings and cells of approximately equal area (in a close analogy to a disco ball) and calculates the emergent spectra for each cell which is visible to the observer (cos theta > 0)
    - while the FORTRAN plumbing for diskoball is in place/tested, the spigot on the python side is not ready.
- Use the TraceStokes driver
  - `'stoktra'`
  - Traces all Stokes vectors (+ continuum) through the atmosphere at a single wavelength point. Analogous to using a “dipstick” to probe the formation location/environment of absorption lines.

Tools currently available in the *nefarious-funicular* github package are:

- **DiffSens**: A utility to make differential sensitivity plots. Differential Sensitivity plots show the partial derivatives of different lines to changes in Teff, log g, and magnetic field strength.
- **CounterPoint**: A utility to make manual manipulation of synthetic PRT spectra tractable. CounterPoint makes it simple to plot synthetic spectra produced by MoogStokes over an observed spectrum.

MoogPy Installation Instructions:

1. Make sure you have the following things installed on your system:
   f2py – type f2py in a terminal, ensure you do not get a command-not-found error
   gfortran – type gfortran in a terminal, ensure you do not get a command-not-found error
   ATLAS/LAPACK – see if some of the following libraries exist in /usr/lib64/atlas

libatlas.so
liblapack.so
libf77blas.so
(I am unsure of which EXACT library is needed, as it seems to vary with your system architecture and operating system. In any case, if you do not have the correct library, f2py will complain and tell you what exact libraries you need)

python > 2.7 (NOT the version provided by scisoft. If your default python is from scisoft, most likely you will need to install a new version of python directly from the website. Punt to IT/helpdesk here if you don't have the required permissions.)
ipython (not required, but recommended)
The following python packages: (Open python and type import <package_name>, ensure that no package-not-found errors occur)
pyfits
numpy
scipy

2. Download the .zip file from the MoogPy github repository:

   https://github.com/soylentdeen/MoogPy

3. Unzip the .zip file. Be aware of the fact that the .zip file will usually unzip to MoogPy-master, and not MoogPy. You can either rename the directory to MoogPy-master to MoogPy, or just adjust all your other paths accordingly.

   The MoogPy directory contains the following files/directories:
   TBC

4. Edit by hand the location of the Moog source code in line 21 in the MoogStokessilent.f

```
moogpath =
. '/absolute/path/to/MoogSource/directory/'
```

5. Download the Absorption Line, Atmosphere, and Observed Spectra databases from:

   http://www.mpia.de/~deen/Software/MoogPy/MoogPy.html

   Put the three tarballs in your favorite directory. Unzip the three tarballs here. This process should result in three directories: (AbsorptionLines, Atmospheres, and ObservedSpectra).

6. Unfortunately, in the halcyon days of FORTRAN, file names and paths were never foreseen to be longer than 80 characters. In the fast and loose software culture of today, the combination of descriptive file names and directory structures can easily make the length of an absolute path to a file longer than this limit. In order to get around this restriction, we can use symbolic links to shorten the absolute path of input and output files. To do this, we need to make a symbolic link to the location of the data.

```
/home/user > ln -s /absolute/path/to/MoogPyData MoogPyData
```

This will create the following symbolic link:

```
/home/user/MoogPyData → /absolute/path/to/MoogPyData/
```

7. Ensure the following environment variables are set to proper values
PYTHONPATH – should include the MoogTools directory inside MoogPy
MOOGPYDATAPATH – should point to the MoogPyData symbolic link

   if you are using BASH, putting the following lines in $HOME/.bashrc will do the trick

```
export PYTHONPATH=/other/python/paths/:/path/to/MoogPy/MoogTools/
export MOOGPYDATAPATH=/home/user/MoogPyData/
```

8. If you are on a Mac running scisoft, you will need to unset the following variables

```
unset F77
unset LDFLAGS
unset F2C
```

   If you are unsure if you have scisoft, open a terminal and type:

```
user@host ~> which python
```

   If the returned path contains some variant of "scisoft", chances are good that you are running scisoft.

   The best way to do this is to place the above commands at the beginning of the `./MoogSource/recompileMoogStokes.sh` file

9. Change the following path in the ./MoogSource/recompileMoogStokes.sh file:

```
-L/usr/lib64/atlas/
```

   to reflect the actual location of the atlas library on your system.

   (N.B. On Dan's MacBook 10.8.5, these libraries are in /usr/lib/

   On my OpenSuse linux distribution, I need to include the following libraries:
   - -llapack
   - -lf77blas
   - -latlas
   - -lcblas
   - -lsatlas
   - -ltatlas

   However, on Dan's MacBook, only the -llapack library is necessary. Indeed, the compile will fail if you include the other -libraries. I have no idea why this is. If the compile script complains about missing libraries, try adding them. The format is liblapack.so = -llapack, libf77blas.so = -lf77blas, etc...

If you are also installing on a Mac, give the recompileMoogStokes_forDan.sh a try. It may not work right out of the box, but it may be closer to your system than the one developed for my Linux box.)

10. If necessary (i.e. if you get errors about missing compilers), change the FORTRAN compiler specified in ./MoogSource/recompileMoogStokes.sh file:

    --fcompiler=gnu95 (or gfortran or whatever is appropriate for your system)

    (N.B. This option is not included in the default recompileMoogStokes.sh file, but it should inserted before/after the –f77flags=-fcheck=all option. On Dan's MacBook 10.8.5, the --fcompiler variable should be set to gnu95)

11. run ./recompileMoogStokes.sh

    If the script does not run, make sure you have the ./ in the command (./ is a way to indicate to the command shell to explicitly look in the current directory). Also, ensure that the command is executable (if not, run `chmod +x ./recompileMoogStokes.sh`)

    This script should run f2py and do the wrapping of the F77 in python. If the routine fails with an error, this is bad, and means that MoogStokesPy did not compile/install. Unless there is a subtle difference in your compiler of which I am unaware, the error is most likely a linking error. Check to ensure that the libraries in the recompile script are indeed pointing to the correct locations and that the library exists. If a given library does not exist on your system, it may simply be subsumed in a different library.

12. Go to the MoogPy base directory. Open up ipython

    ```
    username@host: MoogPy> ipython
    ```

13. Run StokesExample.py

    ```
    In [1]: run StokesExample.py
    ```

    If all goes well, you should see MoogStokes start printing out wavelengths in the ipython terminal. After the spectrum is calculated, you should see a spectrum displayed in a Matplotlib figure. If you see the wavelengths in the terminal but not the spectrum, check to make sure you have the correct backend installed for Matplotlib.

14. Profit.

To describe what is going on under the hood, let's look at the StokesExample.py:

```
import MoogTools                     # MoogPy wrapper library
import matplotlib.pyplot as pyplot   # Python plotting library

fig = pyplot.figure(0)
fig.clear()
```

```
ax = fig.add_axes([0.1, 0.1, 0.8, 0.8])

# Configuration file which contains all parameters for MoogStokes
moogPyConfigFile = 'StokesExample.cfg'

# Instantiate the MoogStokes object
Moog = MoogTools.MoogStokesSpectrum(moogPyConfigFile, fileBase = 'ex-
ample')
# Run MoogStokes, extract spectrum into Python
wavelength, flux = Moog.run(save=True)

# Plot emergent spectrum
ax.plot(wavelength, flux)
fig.show()
```

As you can see, the interaction between the user and MoogStokes is limited to two lines, one to instantiate the MoogStokes object, and one to trigger the execution of the program. The MoogTools object handles the interface to the program for the user.

The user can specify the detailed parameters to control MoogStokes in the configuration file passed to the MoogStokes object. Here is the contents of the StokesExample.cfg file referenced in the above example:

```
# Global Parameters
Bfield = 2.0     # kilo-Gauss
Teff = 3600      # Kelvin
logg = 4.0       # cgs dex
vsini = 5.0      # km/sec
grid = True       # Whether or not Moog is restricted to the
                  # parameters covered by the models

# Wavelength bounds of synthesis
wlStart = 22020
wlStop = 22120

# Location to write the lists of lines extracted from the
# Master line list (wlStart < wl < wlStop)
Strong_FileName = ./MoogSandbox/strong.lines
Weak_FileName = ./MoogSandbox/weak.lines
deltav = 0.01

# Atomic and Molecular line files for generating Moog line lists
strong_file = AbsorptionLines/VALD/lines.strong
VALD_file = AbsorptionLines/VALD/vald_master.txt
molecules = AbsorptionLines/Molecular/
```

```
gf_file = AbsorptionLines/corrections/corrected_lines.dat
doMolecules = False

# Parameters for the Moog .par file
moog_Parameters = MoogStokesPar.cfg

# Parameters for saving the output
outbase = StokesExample
outdir = ./Output/
```

The first section of the configuration file tells MoogStokes what parameters to put into the sythesis. Teff and logg are used to select the model atmosphere. Currently you are restricted to running models on the grid of model atmospheres. For the MARCS models, the grid is structured in the following manner:

- 2500 K <= Teff <= 4000 K – Temperature steps every 100 K, log g spans from 3.0 – 5.5 in steps of 0.5 dex
- 4000 <= Teff <= 6000 K – Temperature steps every 250 K, log g spans from 3.0 to 5.0 in steps of 0.5 dex

While the options for Teff and logg are currently discrete, the user is free to select any positive value for vsini and B.

The second section of the configuration file tells MoogStokes what wavelength range of the spectrum to synthesize. The units of the starting and stopping wavelengths are currently in Angstroms. You are in principle free to synthesize as large a spectral region as you would like, but bear in mind these two caveats:

- Large (> 500 Angstrom) spectral regions can take a while to synthesize
- The adaptive-mesh wavelength grid is constructed
- The adaptive-mesh wavelength grid routine has a limit of 50000 "contributing" and "strong" lines each. A "contributing" line is a line which is more than 1% deep at line center, and a "strong" line is a line which is more than 10% deep at line center. Your line list can contain more lines than this if some of them (due to abundance or ionization effects) simply do not contribute significantly to the opacity.

The last section of the configuration file tells MoogPy where to save the final version of the spectrum. The string in "outdir" is the directory in which the synthesized .fits file is stored, while the "outbase" is the base file name out of which the output file name is created. The format of a saved file is the following:

```
/outdir/outbase_TWWWW_GX.XX_BY.YY_VZ.Z.fits
```

where WWWW = Temperature, X.XX = log g, Y.YY = B field, and Z.Z = v sin(i)

How to install the suite of programs which use MoogPy:

1. Download the .zip file of the repository from:

   http://www.github.com/soylentdeen/nefarious-funicular/

(The download link can be found in the "Download ZIP" button in the lower right)

2. Unzip it, and enter the nefarious-funicular directory (you may have to rename it from nefarious-funicular-master). You should find the following two sub-directories
   - **CounterPoint**: - allows you to over-plot synthetic spectra with observed spectra more or less on-the-fly, adjust wavelength and continuum shifts, apply veiling
   - **DiffSpec**: - allows you to produce a "differential sensitivity plot" for a desired spectral region, set of fiducial stellar parameters, and delta steps. Relative to the continuum.

CounterPoint:

1. Start three terminals
2. In Terminal #1, enter the CounterPoint directory, start ipython:

```
user@host ~> cd /path/to/CounterPoint/
user@host CounterPoint>ipython
```

3. In Terminal #2, enter the Orchestra directory, start ipython:

```
user@host ~> cd /path/to/CounterPoint/Orchestra
user@host Orchestra>ipython
```

4. In Terminal #1, run python program "cp.py" and pass the desired CounterPoint configuration file as a variable

```
In [1]: run cp.py KbandTi.IGRINS
```

The configuration file contains the following keywords:

```
# Contents of KbandTi.IGRINS
watched_dir = ./Output/                      # Directory to watch for new spectra
observed = ./Observed/TWHydra_IGRINS.fits    # Path to observed spectra
wlStart = 22200                              # Starting Wavelength
wlStop = 22300                               # Stopping Wavelength
wlOffset = 40.0                              # Wavelength shift in Observed Spectrum
continuum = 0.97                             # Scaling factor for observed spectra
veiling = 0.1                                # Amount of Veiling
resolving_power = 40000.0                    # Convolution R for synthetic spectra
```

This should plot an observed spectrum and any available synthetic spectra. The program will give the following options:

`s(e)lect plotted spectra` – looks in the watched_dir for any new files, allows user to toggle which spectra to plot along with the observed.
`(w)avelength shift` – Shifts the observed spectrum in wavelength (in units of Angstroms)
`(c)ontinuum` – Scales the observed flux by an input value
`(s)ave plot` – Saves the plot as a .png file.

`(q)uit` – Quits the program.

5. If no synthetic spectra overlap the wavelength range specified in the CounterPoint configuration file, only the observed spectrum will be plotted. If you would like to generate a new spectrum for plotting, do the following: In Terminal #3, enter Orchestra directory, edit spectrum.cfg:

   ```
   user@host ~> cd /path/to/CounterPoint/Orchestra
   user@host Orchestra> vim spectrum.cfg
   ```

   Change the wavelength region to synthesize in the Orchestra/spectrum.cfg file to overlap the wavelength plotting range specified in the CounterPoint configuration file. Also change the model atmosphere parameters (Teff, logg) and magnetic field (Bfield) to your desired settings.

   Next, go to Terminal #2, and run the generateSpectrum.py file:

   ```
   In [1]: run generateSpectrum.py
   ```

   This will generate a new spectrum, and save it in the directory specified by the **outdir** keyword in Orchestra/spectrum.cfg.

6. Now, go back to Terminal #1 and choose 'e'. The program will re-scan the monitored directory specified by the **watched_dir** in the CounterPoint configuration file, and list all synthetic spectra which overlap the requested wavelength region. You should see the new spectrum generated by MoogStokes as an option to plot. If not, ensure that the **watched_dir** and **outdir** point to the same directory. Press enter. The plot should now show your newly synthesized spectrum in the top-half of the figure, and difference between the observed and synthesized in the lower half of the figure.

DiffSens:

DiffSens is a suite of routines that allows a user to generate differential sensitivity plots. The differential sensitivity of a spectrum is a partial derivative (in units of the continuum level) of the spectrum when changing one or another of the input parameters (Teff, log g, or magnetic field strength). The DiffSens sub-directory is on the same hierarchical level as the CounterPoint directory. To generate a differential sensitivity plot of your own, follow this procedure:

1. Open a terminal, go to the DiffSens directory.

   ```
   user@host ~> cd /path/to/nefarious-funicular/DiffSens/
   ```

2. Rename the example.diffsens configuration file to a file name of your choosing. Edit this new file. You will notice that the contents of this DiffSens configuration file are very similar to the StokesExample.cfg configuration file described earlier in the MoogPy installation instructions, but with a few extra lines. The lines which are in common with the StokesExample.cfg file define the “fiducial” model, while the extra lines (with the prefix “delta_”) are used to define the size of the one-off departures from the fiducial model used to calculate the partial derivatives. The “delta_resolution” keyword defines the resolving power at which to process the data.

3. Once you have defined the parameters for your differential sensitivity plot, save the configuration file, and start ipython:

   ```
   user@host DiffSens> ipython
   ```

4. From within the ipython environment, run the program “generateDiffSpectra.py” and pass the configuration file as a parameter:

   ```
   In [1]: run generateDiffSpectra.py example.diffsens
   ```

   The python script will automatically generate individual MoogStokesPy configuration files for each of the 7 spectra needed to calculate the differential sensitivity:
   -fiducial
   -plus T, plus G, plus B
   -minus T, minus G, minus B

   The python script then sequentially spawns a MoogStokes instance for each parameter file, synthesizing 7 separate spectra, convolving each to the specified resolution, and then interpolating all spectra to a common wavelength scale. The partial derivative for a parameter is then calculated by:

   dF(lambda)/dX = (plus_X(lambda) – minus_X(lambda))/2.0

   The python script then produces, displays, and saves a differential sensitivity plot.

   The name of the sensitivity plot is constructed from the **outbase** and **delta_resolution** keywords.