Jan. 21, 2022

Installing and making sure that Hazel2 works

For most of our exercises during the “HeI diagnostics” school, we will be using the HAZEL code (from HAnle and ZEeman Light) written by Andres Asensio Ramos and described in the paper: Asensio Ramos et al., 2008, ApJ 683, 542. In particular, we’ll be using the Hazel 2.0 version, written in python.

All of the following has been taken from the extensive documentation provided by Andres:

https://aasensio.github.io/hazel2/started/installation.html

We note that some small things are likely to be different, e.g., the version of python compared to the guide, etc. But these are (or should be) minor differences. If you encounter any problems during installation and testing, you can ask us for help at the discord server, or during the office hours, as indicated in the email.

We (and the author of the code) suggest using so-called python environments for running Hazel2. An environment is like a small, local, mini-filesystem, where you can install specific versions of packages. It requires having the so-called conda, which is a python (and not only python!) package / environment manager.

First, you will need anaconda, or miniconda installed. This will work in Linux, MacOS, as well as in windows (more details below). After installing anaconda / miniconda, go through the following steps:

MacOS:

To the best of our knowledge, it is impossible to run hazel2 on MacOS machines without first installing **xcode** or at least the **command line tools.** These packages allow you to run compilation commands in your terminal, and without them you will not be able to build hazel2. If you don’t have them already, **command line tools** are lighter and faster to install, so before everything else, start by installing that, following this link:

https://mac-how-to.gadgethacks.com/how-to/install-command-line-developer-tools-without-xcode-0168115/

If you do not have privileges to do that (e.g. you are using your institute's computer), it is a good time to get in touch with them now and ask them to do that for you.

Provided that went fine, open your terminal and do the following:

Create an environment for running Hazel2 (the python version does not matter as long as it is a python 3; choose whatever is the latest), activate it and install the necessary packages (the 3rd and 4th line are one big command):

```
conda create --name hazel_env python=3.9
conda activate hazel_env
conda install -c conda-forge cython numpy h5py tqdm scipy astropy
mpi4py configobj gfortran_osx-64 clang asciitree
conda install -c conda-forge git ipython jupyter matplotlib
```

Important: Sometimes the "-" signs will get copied improperly, resulting in errors. If you get any weird behavior, just type out that argument (e.g. --name) instead of copying it. Also clang might have to be replaced with clang_osx-64, or clangxx_osx-64.

Linux and Windows:

If you are a windows user, it seems that the easiest path is to install the so called Linux Subsystem for Windows (WSL), and follow the steps below (your subsystem now behaves like an independent Linux computer). To install WSL, you can follow, for example, this guide:

https://docs.microsoft.com/en-us/windows/wsl/install

The guide does mention multiple possible distributions of Linux. It is simplest to just use the most popular one - Ubuntu. After that, you want to open your Ubuntu installation and follow these steps:

First, install conda or miniconda in your Linux (or WSL):

https://docs.conda.io/projects/conda/en/latest/user-guide/install/linux.html

After that, in a terminal, execute the following (3rd, 4th, 5th line are one big command):

```
conda create --name hazel_env python=3.9
conda activate hazel_env
```

```
conda install -c conda-forge cython numpy h5py tqdm scipy
astropy mpich mpi4py configobj gfortran_linux-64 gcc_linux-64
asciitree
conda install -c conda-forge git ipython jupyter matplotlib
```

Installing and testing Hazel2 (same for all three operating systems)

Finally, download and install Hazel2! Open a terminal, go to a folder where you keep your codes and do the following. A lot of compiling (as well as warnings) will follow, but do not worry. We are looking mostly for errors. If these happen, make a note and come to us!

```
conda activate hazel_env
git clone https://github.com/aasensio/hazel2
cd hazel2
python setup.py install
```

(If you continue using hazel2, you can update to a newer version at any point using:
`git pull`
`python setup.py install )`

To finally check if everything is ok, we suggest:

```
cd ~/
ipython
```

And after ipython is running, do:

```
import hazel
```

If this works fine, it means the code is properly installed! (**Note:** in your home folder, or wherever you are when you run ipython and try to import hazel, there must not be another folder named hazel, or python will simply import its contents whatever they might be, instead of the hazel2 package).

To try out if notebooks work and if they can properly see hazel2, first exit the ipython session:

```
exit()
```

Then you can open a notebook, by typing:

```
jupyter notebook
```

Your browser will appear, showing the contents of the directory you are in. Above, to the right, from the list of the files, click “new” and click “python3”, it will open a new notebook. Copy and paste the following in the cell that appeared:

```
import hazel
import matplotlib.pyplot as plt
mod = hazel.Model(working_mode='synthesis')
mod.add_spectral({'Name': 'spec1', 'Wavelength': [10826, 10833, 150], 'topology': 'ch1',
    'LOS': [0.0,0.0,90.0], 'Boundary condition': [1.0,0.0,0.0,0.0]})
mod.add_chromosphere({'Name': 'ch1', 'Spectral region': 'spec1', 'Height': 3.0, 'Line':
'10830', 'Wavelength': [10826, 10833]})
mod.setup()
mod.synthesize()

f, ax = plt.subplots(nrows=2, ncols=2)
ax = ax.flatten()

for j in range(5):
    # Vector of parameters are (Bx,By,Bztau,v,deltav,beta,a) and then the ff
    mod.atmospheres['ch1'].set_parameters([0.0,0.0,100.0*j,1.0,0.0,8.0,1.0,0.0],1.0)
    mod.synthesize()

    for i in range(4):
        ax[i].plot(mod.spectrum['spec1'].stokes[i,:])
plt.show()
```

And then press shift+Enter. This ‘executes’ the cell. After a while, you should see something like this. If you do, that means your Hazel2 works. Enjoy!

jupyter Untitled Last Checkpoint: a few seconds ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

Run Code

In [1]:
```
import hazel
import matplotlib.pyplot as plt
mod = hazel.Model(working_mode='synthesis')
mod.add_spectral({'Name': 'spec1', 'Wavelength': [10826, 10833, 150], 'topology': 'ch1',
    'LOS': [0.0,0.0,90.0], 'Boundary condition': [1.0,0.0,0.0,0.0]})
mod.add_chromosphere({'Name': 'ch1', 'Spectral region': 'spec1', 'Height': 3.0, 'Line': '10830', 'Wavelength': [10826, 10833]})
mod.setup()
mod.synthesize()

f, ax = plt.subplots(nrows=2, ncols=2)
ax = ax.flatten()

for j in range(5):
    # Vector of parameters are (Bx,By,Bztau,v,deltav,beta,a) and then the ff
    mod.atmospheres['ch1'].set_parameters([0.0,0.0,100.0*j,1.0,0.0,8.0,1.0,0.0],1.0)
    mod.synthesize()

    for i in range(4):
        ax[i].plot(mod.spectrum['spec1'].stokes[i,:])
plt.show()
```