Using MATLAB with Jupyter Notebook

SeHyoun Ahn

2018-09-04

1 Using MATLAB with Jupyter Notebook

(pdf of this post is available: here)

This is a meta-post about blogging. I use Jupyter Notebook with Pelican to create blog posts. I might convert my codes to some other language (Julia) in the future, but I already have significant codebase built up in MATLAB that the up-front cost of making the jump is large. However, with MATLAB-Python integration it is not too painful to use them together, and using this toolchain works better (for me) than using the publish function I have used before to generate replication pages.

1.1 Installing MATLAB engine

First, to be able to call MATLAB from python, one needs to install the matlab engine. One can read about the MATLAB-python integration at Mathworks's documentation page.

1.1.1 Aside

The <setup.py> provided with MATLAB (2017b) does not find the proper location for installing the proper files if you are using Anaconda instead of MATLAB. However, since the setup file just finds python location and places a folder there, patching it to work with Anaconda is simple.

```
1. Find the location of the matlab module folder by looking at the output from running <python s
2. Copy the folder and place it into anaconda3/lib/python3.x/site-packages
```

This should be enough to get matlab engine to function properly with Anaconda.

1.2 Magic Method for Jupyter Notebook

Now, one can just follow the documentation from MATLAB to call MATLAB from Python, but using it at this setup is annoying from Python side as the output is sent to the stdout of the OS (command line calling jupyter notebook in linux). However, this is easy to fix by declaring a magic method within Jupyter Notebook. The function that I ususally setup is

```
In [1]: import matlab.engine
        import io
        from IPython.core.magic import register_cell_magic
        ip = get_ipython()
```

```
        out = io.StringIO()
        err = io.StringIO()

        # Setup matlab cell magic #
        @register_cell_magic
        def matlab_magic(line,cell):
            out.truncate(0)
            out.seek(0)
            err.truncate(0)
            err.truncate(0)
            raw = '''{line}.eval("""{cell}""", nargout=0, stdout=out, stderr=err)'''
            ip.run_cell(raw.format(line=line, cell=cell))
            print(out.getvalue())
            print(err.getvalue())
```

This function defines the magic method %%matlab_magic so that one can call MATLAB without having to call through the "engine" syntax. The general syntax of a magic line is

```
%%matlab_magic name_of_matlab_engine

any matlab codes
```

This is basically all that is necessary to call MATLAB from Jupyter Notebook. To see that it works properly, let's see a few examples.

First we start an instance of MATLAB.

```
In [2]: eng = matlab.engine.start_matlab()
```

We can define a variable and check the value following the same syntax as in MATLAB. In fact, one can just consider anything one runs within matlab_magic to be ran within MATLAB.

```
In [3]: %%matlab_magic eng

        x = 5;
        disp(x);
        x
```

```
     5


x =

     5
```

Most functions work as expected and one can make plots as well, and it will call up a matlab figure. However, there is no automatic method to make the plots inline unlike what you can do with matplotlib.

```
In [4]: %%matlab_magic eng

        y = linspace(0,1,10);
        disp(y);
        plot(y);

  Columns 1 through 7

         0    0.1111    0.2222    0.3333    0.4444    0.5556    0.6667

  Columns 8 through 10

    0.7778    0.8889    1.0000
```

One can however, save the figure and attach the image to Jupyter Notebook without much problem via

```
In [5]: %%matlab_magic eng

        saveas(gcf, 'using_matlab_with_jupyter_notebook_figure1.png');
```

And attach figure following the syntax

```
![alt text here](./using_matlab_with_jupyter_notebook_figure1.png)
```

We can pretty much use Jupyter Notebook features with underlying engine of MATLAB with this setup now.

1.3 Make MATLAB Magic Persistent

One can also make the magic function automatic to all jupyter notebook by update the startup file located at <~/.ipython/profile_default/startup/start.ipy> in linux (for windows or mac OSes, search for corresponding locations).

One can just place the script given above in In [1] into the file and save. MATLAB magic should be ready with all new jupyter notebook at startup.

1.4 Good-Bye

Finally, we end the post by closing the MATLAB engine via

```
In [6]: eng.quit()
```

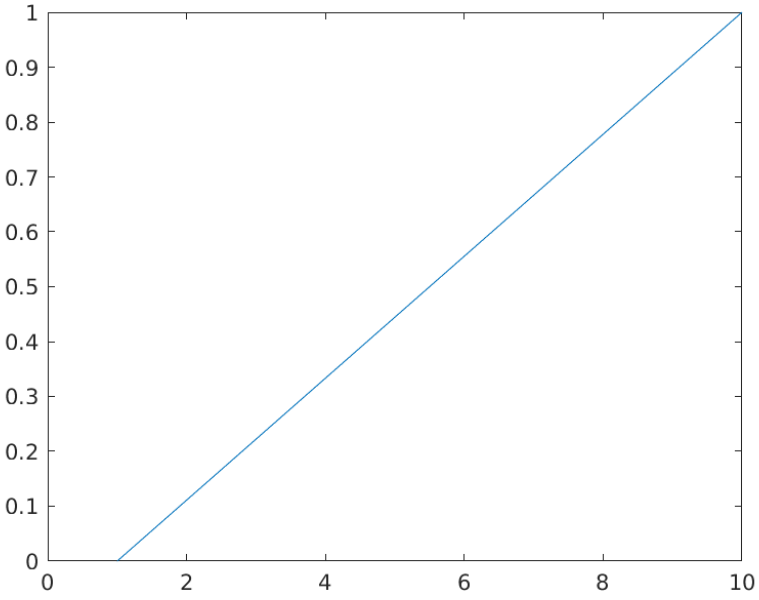

Figure 1