Integrating various kernels into Jupyter

Introduction

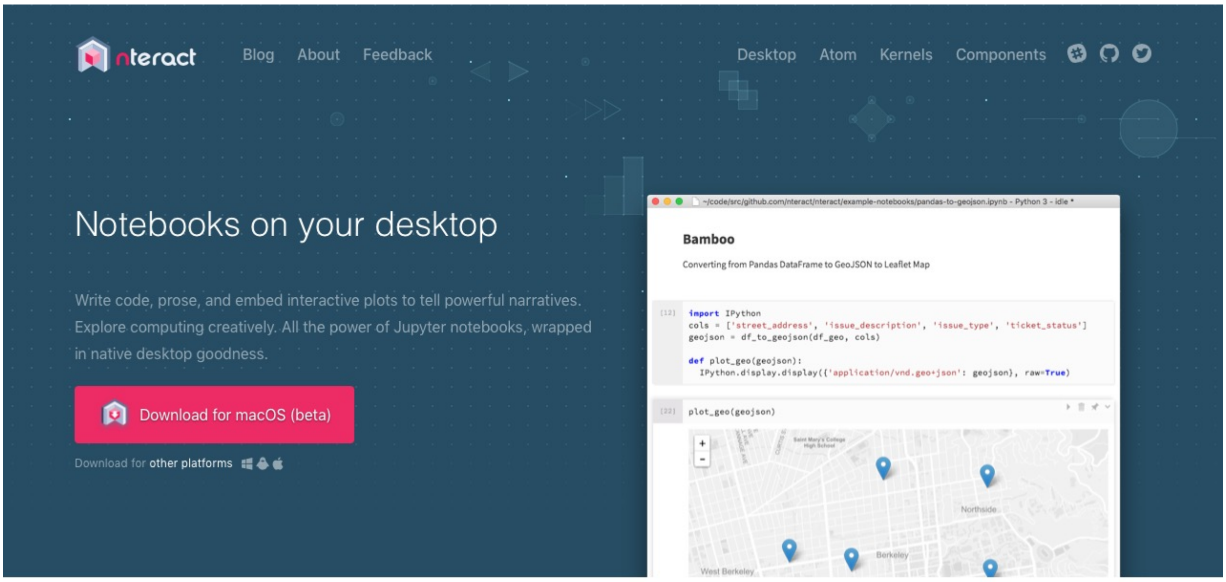
Jupyter notebooks provides a powerful tool to propagate our works. They can be used to integrate various kernels into one interface: Matlab, Mathematica and Python for example. Moreover, all the writings can be performed in `Markdown` which is based on `html` and $\LaTeX$ approach to writing.

This is `Markdown`, and as you see is very *nice* and **handy**. An inline formula here a_n, and an eqution below ↓

$$\frac{\sqrt{2}\sqrt{z}j_{\nu-\frac{1}{2}}(z)}{\sqrt{\pi}}.$$

To download the nicest packages you need to install Anaconda with the latest version of *Phyton* (3.5 or above). This will give you the first and original kernel of the conda enviroment. You will need to create a user profile to make it work, and share notebooks with other people.

Then you need to download nteract, which let's you maniplate several kernels from a very nice notebook *á lá Mathematica* interface. You will need to create a GitHub account to share the notebooks from *nteract* (you already did it for *Mathlab*). Read **this** to see all the powerfull ways of showing off your scientific document.

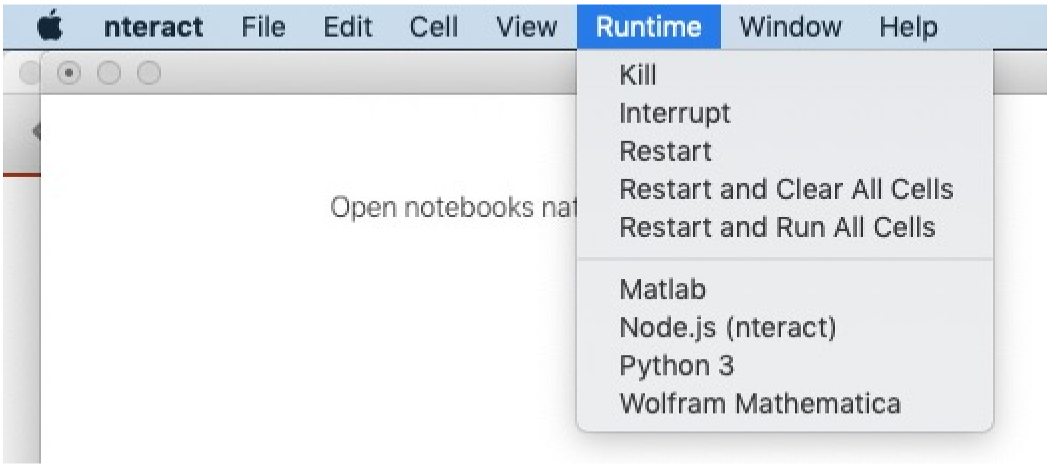


Additional Kernels (Matlab and Mathematica)

The *Matlab kernel* for Jupyter needs to downloaded. Before running this setup using `$pip install matlab_kernel` you need to have installed the Matlab engine for Python.

If everything ran smoothly you should be able to switch to *Matlab kernel*. In *Phyton* versions greater than *3.6*, an error will occur when loading the Jupyter kernel. Although it will solved some time in the near future, a work around exists replacing `line 251` in the `kernel.py` file found at `/Users/<user>/anaconda3/lib/python3.7/site-packages/matlab_kernel/` by

```
251      future = self._matlab.eval(code, **{ 'nargout': 0, 'async': True}).
```



Once everything is settle down, you can run it

[1]
```
disp('hello from MATLAB')
```

```
hello from MATLAB
```

On the other hand, Wolfram has introduced a *Link* between **Mathematica** and Python. Your kernel should be set to Python. Then,

[1]
```
from wolframclient.evaluation import WolframLanguageSession
```

[2]
```
from wolframclient.language import *
```

[3]
```
session = WolframLanguageSession()
```

This three lines are required to create a Mathematica Session in Python.

[4]
```
session.evaluate(wlexpr('Range[5]'))
```

```
[1, 2, 3, 4, 5]
```

A basic symbolic evaluatio looks like this.

```
[7] session.evaluate('Expand[(x+1)^2] // MatrixForm')
```

```
MatrixForm[Plus[1, Times[2, Global`x], Power[Global`x, 2]]]
```

which is not so useful... imagine a BesselJ assymptotic expansion:

```
[99] session.evaluate('Series[BesselJ[nu, x], {x, Infinity, 2}]')
```

```
Plus[Times[Cos[SeriesData[Global`x, DirectedInfinity[1], << 2 >>, 3,
1]], SeriesData[Global`x, DirectedInfinity[1], << 2 >>, 5, 2]],
Times[SeriesData[Global`x, DirectedInfinity[1], << 2 >>, 5, 2],
Sin[SeriesData[Global`x, DirectedInfinity[1], << 2 >>, 3, 1]]]]
```

It becomes unreadeable. An `ExporString` function must be called, but still

```
[100] session.evaluate('ExportString[Expand[(1+x)^2], "TeXFragment"]')
```

```
'\\[1+2 x+x^2\\]\n\n'
```

It is almos $\LaTeX$. But you need to replace `'\\[` and `\\]\n\n'` by `$$`

$$1 + 2x + x^2$$

to make it friendly. For more long calculations like this

```
[119] session.evaluate('ExportString[Series[BesselJ[n, x], {x,
Infinity, 2}],"TeXFragment"]')
```

```
'\\[\\text{Cos}\\left[-x+\\frac{1}{4} (\\pi +2 n \\pi
)+O\\left[\\frac{1}{x}\\right]^3\\right] \\left(\\sqrt{\\frac{2}{\\pi }}
\\sqrt{\\frac{1}{x}}+O\\left[\\frac{1}
{x}\\right]^{5/2}\\right)+\\left(\\frac{\\left(-1+4\\nn^2\\right)
\\left(\\frac{1}{x}\\right)^{3/2}}{4 \\sqrt{2 \\pi }}+O\\left[\\frac{1}
{x}\\right]^{5/2}\\right) \\text{Sin}\\left[-x+\\frac{1}{4} (\\pi +2 n
\\pi )+O\\left[\\frac{1}{x}\\right]^3\\right]\\]\n\n'
```

Then just by *replacing* every \\ by a single \ and the ending and beginging makes it difficult and time consuming just for a simple result

$$\left(\frac{(4n^2-1)\left(\frac{1}{x}\right)^{3/2}}{4\sqrt{2\pi}}+O\left(\left(\frac{1}{x}\right)^{5/2}\right)\right)\sin\left(-x+\frac{1}{4}(2\pi n+\pi)+O\left(\left(\frac{1}{x}\right)^{3}\right)\right)+$$
$$+\left(\sqrt{\frac{2}{\pi}}\sqrt{\frac{1}{x}}+O\left(\left(\frac{1}{x}\right)^{5/2}\right)\right)\cos\left(-x+\frac{1}{4}(2\pi n+\pi)+O\left(\left(\frac{1}{x}\right)^{3}\right)\right).$$

This is too convoluted, and a simple approach is abscent from *Wolfram* side. It is best to install the `wolfram kernel` from Mauricio Matera. You need to download it and run (**Windows and Linux**)

```
$ python setup.py install --mma-exec <Mathematica executable>
```

On **MacOS** you need to put the script `wmath` into `/Applications/Mathematica.app/Contents/MacOS/` and then `$sudo chmod 755 wmath`. Once this is done, run

```
$ python setup.py install --mma-exec
/Applications/Mathematica.app/Contents/MacOS/wmath
```

A simple command will look like this

```
[1]  Series[Exp[x],{x,0,3}]
```

```
SeriesData[x, 0, {1, 1, 1/2, 1/6}, 0, 4, 1]
```

If you want to have the result expressed in $\LaTeX$ form just add `//Matrix Form` at the end like this

```
[1]  FunctionExpand[Factorial2[n], Assumptions -> n \[Element]
     Integers] //MatrixForm
```

$$2^{\frac{n}{2}+\frac{1}{4}(1-\cos(\pi n))}\pi^{\frac{1}{4}(\cos(\pi n)-1)}\Gamma\left(\frac{n}{2}+1\right)$$

```
[2] Series[BesselJ[n,x],{x,Infinity,1}] //MatrixForm
```

$$\left(\frac{(4n^2-1)\left(\frac{1}{x}\right)^{3/2}}{4\sqrt{2\pi}}+O\left(\left(\frac{1}{x}\right)^2\right)\right)\sin\left(-x+\frac{1}{4}(2\pi n+\pi)+O\left(\left(\frac{1}{x}\right)^2\right)\right)+\left(\sqrt{\frac{2}{\pi}}\sqrt{\frac{1}{x}}\right.$$

But if you want just the $\LaTeX$ code ouput use `TexForm` like this

```
[4] TeXForm[Series[BesselJ[n,x],{x,Infinity,1}]]
```

```
\left(\frac{\left(4 n^2-1\right) \left(\frac{1}{x}\right)^{3/2}}{4
\sqrt{2 \pi }}+O\left(\left(\frac{1}{x}\right)^2\right)\right) \sin
\left(-x+\frac{1}{4} (2 \pi  n+\pi )+O\left(\left(\frac{1}
{x}\right)^2\right)\right)+\left(\sqrt{\frac{2}{\pi }} \sqrt{\frac{1}
{x}}+O\left(\left(\frac{1}{x}\right)^{3/2}\right)\right) \cos \left(-
x+\frac{1}{4} (2 \pi  n+\pi )+O\left(\left(\frac{1}
{x}\right)^2\right)\right)
```

Some integrals may contain *kernel messages* about potential problems in the expresion

```
[7] Integrate[
     Exp[I (A Cos[\[Theta]] + B Sin[\[Theta]])], {\[Theta], 0, 2 \
    [Pi]}, Assumptions -> A>0 && B>0] //MatrixForm
```

```
Infinity::indet: StringForm[Infinity::indet /. Messages[Infinity],
HoldForm[0*Infinity]]

Infinity::indet: StringForm[Infinity::indet /. Messages[Infinity],
HoldForm[0*Infinity]]

Infinity::indet: StringForm[Infinity::indet /. Messages[Infinity],
HoldForm[-Infinity + Infinity]]

Infinity::indet: StringForm[Infinity::indet /. Messages[Infinity],
HoldForm[-Infinity + Infinity]]

Infinity::indet: StringForm[Infinity::indet /. Messages[Infinity],
HoldForm[E^ComplexInfinity]]

Infinity::indet: StringForm[Infinity::indet /. Messages[Infinity],
HoldForm[E^ComplexInfinity]]

Infinity::indet: StringForm[Infinity::indet /. Messages[Infinity],
```

```
Infinity::indet: StringForm[Infinity::indet /. Messages[Infinity],
HoldForm[E^ComplexInfinity]]

Infinity::indet: StringForm[Infinity::indet /. Messages[Infinity],
HoldForm[E^ComplexInfinity]]

Infinity::indet: StringForm[Infinity::indet /. Messages[Infinity],
HoldForm[0*Infinity]]

Infinity::indet: StringForm[Infinity::indet /. Messages[Infinity],
HoldForm[0*Infinity]]

Infinity::indet: StringForm[Infinity::indet /. Messages[Infinity],
HoldForm[0*Infinity]]

Infinity::indet: StringForm[Infinity::indet /. Messages[Infinity],
HoldForm[0*Infinity]]

Infinity::indet: StringForm[Infinity::indet /. Messages[Infinity],
HoldForm[0*Infinity]]

Infinity::indet: StringForm[Infinity::indet /. Messages[Infinity],
HoldForm[0*Infinity]]

Infinity::indet: StringForm[Infinity::indet /. Messages[Infinity],
HoldForm[0*Infinity]]

Infinity::indet: StringForm[Infinity::indet /. Messages[Infinity],
HoldForm[0*Infinity]]

Infinity::indet: StringForm[Infinity::indet /. Messages[Infinity],
HoldForm[-Infinity + Infinity]]

Infinity::indet: StringForm[Infinity::indet /. Messages[Infinity],
HoldForm[-Infinity + Infinity]]

Infinity::indet: StringForm[Infinity::indet /. Messages[Infinity],
HoldForm[E^ComplexInfinity]]

Infinity::indet: StringForm[Infinity::indet /. Messages[Infinity],
HoldForm[E^ComplexInfinity]]

Infinity::indet: StringForm[Infinity::indet /. Messages[Infinity],
HoldForm[E^ComplexInfinity]]

Infinity::indet: StringForm[Infinity::indet /. Messages[Infinity],
HoldForm[E^ComplexInfinity]]

Infinity::indet: StringForm[Infinity::indet /. Messages[Infinity],
HoldForm[E^ComplexInfinity]]
```

```
Infinity::indet: StringForm[Infinity::indet /. Messages[Infinity],
HoldForm[0*Infinity]]

Infinity::indet: StringForm[Infinity::indet /. Messages[Infinity],
HoldForm[0*Infinity]]

Infinity::indet: StringForm[Infinity::indet /. Messages[Infinity],
HoldForm[0*Infinity]]

Infinity::indet: StringForm[Infinity::indet /. Messages[Infinity],
HoldForm[0*Infinity]]

Infinity::indet: StringForm[Infinity::indet /. Messages[Infinity],
HoldForm[0*Infinity]]

Infinity::indet: StringForm[Infinity::indet /. Messages[Infinity],
HoldForm[0*Infinity]]
```

$$2\pi J_0\left(\sqrt{A^2+B^2}\right)$$

Some times this has nothing to do with convergence, but a problem in the *wofram kernel* for Jupyter. If nothing is wrong with the convergence you can turn this messages off using

[15]
```
Off[Infinity::indet,MinValue::lpsub,MaxValue::lpsub];
```

[13]
```
Integrate[
 Exp[I (A Cos[\[Theta]] + B Sin[\[Theta]])], {\[Theta], 0, 2 \
[Pi]},Assumptions -> A>0 && B>0] //MatrixForm
```

$$2\pi J_0\left(\sqrt{A^2+B^2}\right)$$

[14]
```
Series[Hypergeometric2F1[a,b,c,x],{x,Infinity,2}]//MatrixForm
```

$$x^{-a-b}\left(x^b\left(\frac{(-1)^{-a}\Gamma(b-a)\Gamma(c)}{\Gamma(b)\Gamma(c-a)}+\frac{(-1)^{-a}a(a-c+1)\Gamma(b-a)\Gamma(c)}{(a-b+1)\Gamma(b)\Gamma(c-a)x}+\frac{(-1)^{-a}a(a+1)(a-c+1)(a-c+2)\Gamma(b-a)\Gamma}{2(a-b+1)(a-b+2)\Gamma(b)\Gamma(c-a)x^2}\right.\right.$$

This must cover all related to the kernels we are comfortable with. Next update for this notebook will be how to manipulate plots.

Some basics IPython

[1]
```
from sympy import *
```

imports everything from sympy

[3]
```
from mpmath import *
```

imports everything from mpath

[2]
```
k, m, n = symbols('k m n', integer=True)
```

[3]
```
f, g, h = symbols('f g h', cls=Function)
```

[4]
```
x, y, z, nu = symbols('x y z nu')
```

[5]
```
from sympy.abc import n,a,b,x
```

[6]
```
init_printing()
```

The code above turns output into $\LaTeX$ commands.

[7]
```
besselj(nu,z).rewrite(jn)
```

$$\frac{\sqrt{2}\sqrt{z}j_{\nu-\frac{1}{2}}\left(z\right)}{\sqrt{\pi}}$$

[10]
```
1/((x+2)*(x+1))
```

$$\frac{1}{(x+1)(x+2)}$$

[11]
```
apart(1/((x+2)*(x+1)))
```

$$-\frac{1}{x+2}+\frac{1}{x+1}$$

Series Expansion

[8]
```
cos(a*x**2).series(x,0,8)
```

$$1-\frac{a^2x^4}{2}+O\left(x^8\right)$$

[9]
```
jacobi(2,a,b,x).series(x,0,2)
```

$$-\frac{1}{2}+x\left(\frac{a^2}{4}+\frac{3a}{4}-\frac{b^2}{4}-\frac{3b}{4}\right)-\frac{b}{8}+\frac{b^2}{8}-\frac{a}{8}-\frac{ab}{4}+\frac{a^2}{8}+O\left(x^2\right)$$

[10]
```
besselj(n,x).series(x,0,2)
```

$$J_n(0)+x\left(\frac{J_{n-1}(0)}{2}-\frac{J_{n+1}(0)}{2}\right)+O\left(x^2\right)$$

[11]
```
series(besselj(n,x),x,0,2)
```

$$J_n(0)+x\left(\frac{J_{n-1}(0)}{2}-\frac{J_{n+1}(0)}{2}\right)+O\left(x^2\right)$$

While the line below cannot be evaluated. For n or any fixed value of it.

[12]
```
besselj(0,x).series(x,oo,2)
```

Let us chech the *package* mpath to see if hypergeometric functions are of any help.

[17]
```
hyp0f1(2,x).series(x,0)
```

NO. This a floating-point package, not for symbolic manipulation.

[13]
```
f = fps(sin(x))
```

[16]
```
integrate(f,(x,0,1))
```

$$- \cos(1) + 1$$

[19]
```
f.truncate()
```

$$x - \frac{x^3}{6} + \frac{x^5}{120} + O\left(x^6\right)$$

[21]
```
g=fps(besselj(1,x))
```