Jupyter notebook custom conversion

Romain Madar

August 2018

Overview

```
COUNTER=0
```

nbconvert latex test

Printing using python

Pyout (and Text Wrapping)

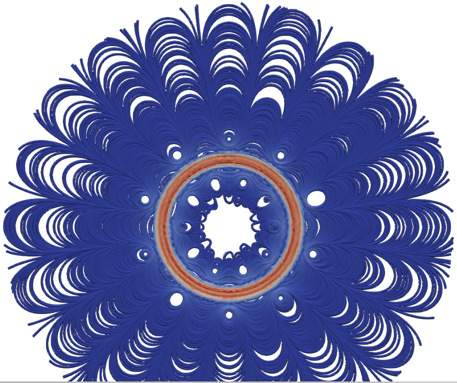
Image and plots

As plain text using markdown

Once exported as markdown and converted to latex/pdf with pandoc, the `{width=60%}` will fix the width of the picture and the `My  legend` will appear as caption:

```
![My legend](figures/magnetostatics_field.png){width=50% #figlabel}
```

gives the result showns in this figure.



Plots produced by the code I

```
#%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(-10,10,300)
y = np.sin(x)
plt.figure(figsize=(4,3),dpi=100)
p=plt.plot(x,y)
```

```
%matplotlib notebook
plt.ioff()
from IPython.display import Markdown

def plt2md(figlabel,figcaption,figsize):
    global COUNTER
    figname = figlabel+'_'+str(COUNTER)+'.png'
    plt.tight_layout()
    plt.savefig(figname)
    plt.savefig(figlabel+'.pdf')
    strMD='![{}]({})'.format(figcaption,figname) +\
        '{' + 'width={} #'.format(figsize) + figlabel + '}'
    COUNTER+=1
    return display(Markdown(strMD))
Markdown('---')
```

```
plt.figure(figsize=(8,3),dpi=150)
plt.plot(x,y)
plt.xlabel('$X_{e}$ [SI]')
plt.ylabel('$Y_{UPDATE12}$ [SI]')
plt2md('myplot','This is a test of how to get proper'+\
       'plot from Jupyter notebook in MD,'+\
       'to be processed using PANDOC','100%')
Markdown('---')
```

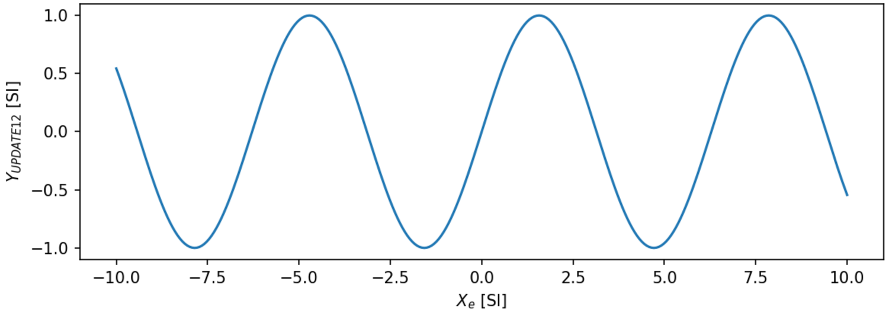


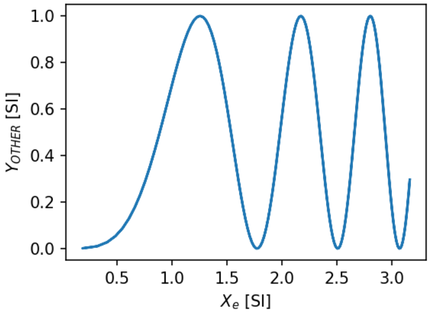
Figure 3: This is a test of how to get properplot from Jupyter notebook in MD,to be processed using PANDOC

We can then refer to a given figure using cross-references like this, obtained with:

```
[like this](#myplot)
```

```
plt.figure(figsize=(4,3),dpi=150)
plt.plot(np.sqrt(np.abs(x)),y**2)
plt.xlabel('$X_{e}$ [SI]')
plt.ylabel('$Y_{OTHER}$ [SI]')
plt2md('myplot2','This is another test of how to get proper'+\
       'plot from Jupyter notebook in MD,'+\
       'to be processed using PANDOC','50%')

display(Markdown('---'))
```



```
plt.plot(x**2,y)
plt2md('myplot3','Adding more plot without re-creating a figure: curves
↪  cumulates','50%')
Markdown('---')
```

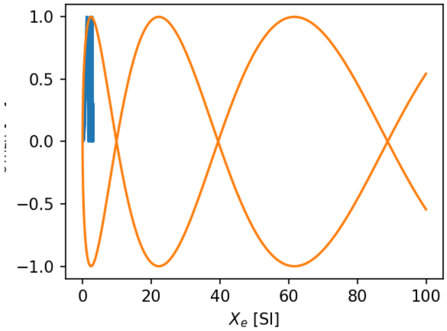


Figure 5: Adding more plot without re-creating a figure: curves cumulates

```
plt.figure(figsize=(4,3),dpi=150)
plt.plot(x**2,y)
plt2md('myplot4','More plot with re-creating a figure: only last
↪  curve','50%')
Markdown('---')
```

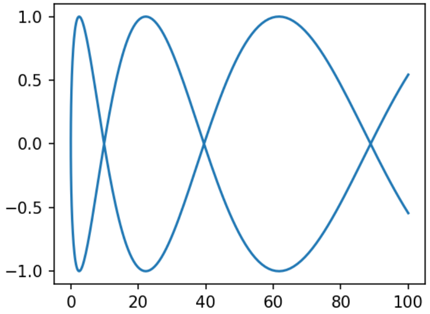


Figure 6: More plot with re-creating a figure: only last curve

Operator Highlighing Check

Tables

Markdown as plain text

First a *markdown* table:

Column 1	Column 2
1	3
~~a~~	b
4	&

Pandas as default and Markdown I

```
import pandas as pd
df=pd.DataFrame(np.random.randn(10,3))

def df2md(df):
    from IPython.display import Markdown
    from tabulate import tabulate
    return display(Markdown( tabulate(df, headers='keys', tablefmt='pipe')
    ↪  ))
```

```
# Default printing is HTML, so it looks good
# on the web but it is not well
# rendered in pdf via ipynb->MB->pdf (using nbconvert
# and pandoc)
df
```

0

1

2

0

Pandas as default and Markdown II

1.705797

-0.246615

-0.831035

1

-0.049958

0.769684

-1.064141

2

0.533581

0.233747

0.109178

3

Pandas as default and Markdown III

-0.004270

-1.729752

-0.026460

4

0.256298

0.836856

0.767965

5

-0.573223

-1.082970

-1.162997

6

Pandas as default and Markdown IV

0.313842

0.656127

-0.190213

7

0.757791

-0.069110

-0.729063

8

0.705909

-1.114225

-0.083281

9

Pandas as default and Markdown V

-0.843936

-0.561475

-0.675126

```
# Markdown printing well rendered
# in MD using nbconvert
df2md(df)
```

	0	1	2
0	1.7058	-0.246615	-0.831035
1	-0.0499584	0.769684	-1.06414
2	0.533581	0.233747	0.109178
3	-0.00426978	-1.72975	-0.0264595
4	0.256298	0.836856	0.767965
5	-0.573223	-1.08297	-1.163
6	0.313842	0.656127	-0.190213
7	0.757791	-0.06911	-0.729063

Pandas as default and Markdown VI

8	0.705909	-1.11422	-0.0832815
9	-0.843936	-0.561475	-0.675126

Sympy output

Line Length