laura

May 27, 2020

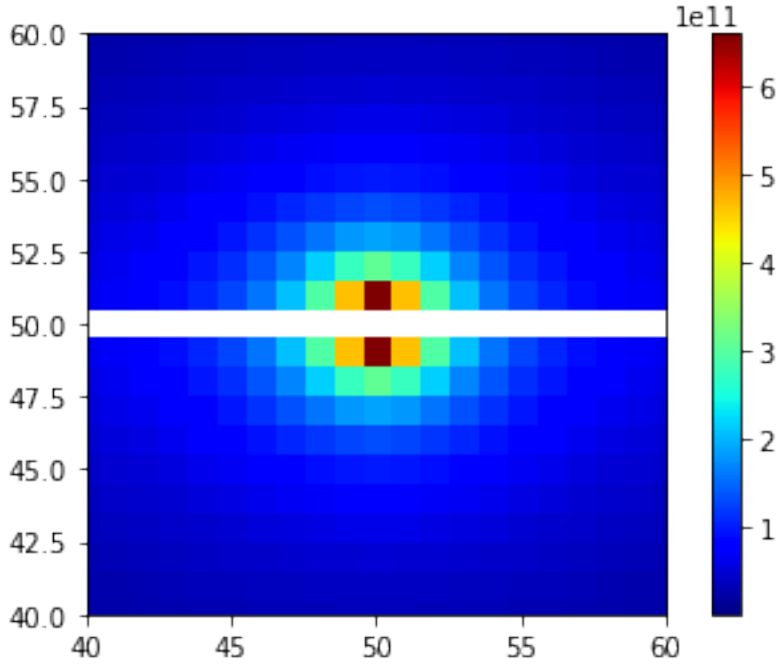
[0]:
```
def simpson_1_3(a,b,n,y1):
 from numpy import linspace,exp
 import math as mt
 h=(b-a)/n
 x=linspace(a,b,n+1)
 y=1/(x**2+y1**2)
 k1,k2=0,0
 if n%2==0:
   n1,n2=n,n-1
 else:
   n1,n2=n-1,n
 for i in range(1,n1,2):
   k1=y[i]+k1
 for i in range(2,n2,2):
   k2=y[i]+k2
 I=(y[0]+y[n]+4*k1+2*k2)*h/3
 return(I)
```

[12]:
```
#programa completo
from numpy import arctan,cos,linspace,zeros,pi,transpose
from pylab import plot,imshow
import matplotlib.pyplot as plt
n=100
E0=8.85418781e-12
x1=linspace(-10,10,n+1)
x2=linspace(-10,10,n+1)

k1=0
A=zeros([len(x1),len(x2)],dtype=float,order='c')
for i in range(0,len(x1)):
  k2=0
  for j in range(0,len(x2)):
    if x2[j]>=0:
      A[k2,k1]=(5/(4*pi*E0))*simpson_1_3(-2,2,n,x2[j])*cos(arctan(x1[i]/
 ↪abs(x2[j])))
      k2=k2+1
    elif x2[j]<0:
```

```
      A[k2,k1]=(5/(4*pi*E0))*simpson_1_3(-2,2,n,x2[j])*cos(arctan(x1[i]/
 ↪abs(x2[j])))
      k2=k2+1
  k1=k1+1

imshow(A,cmap='jet',origin='lower')

plt.xlim(40,60)
plt.ylim(40,60)
plt.colorbar()
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:6: RuntimeWarning:
divide by zero encountered in true_divide

/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:16: RuntimeWarning:
divide by zero encountered in double_scalars
  app.launch_new_instance()
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:16: RuntimeWarning:
invalid value encountered in double_scalars
  app.launch_new_instance()
```

[12]: `<matplotlib.colorbar.Colorbar at 0x7fa7b2eb2e10>`



[0]:
```
!apt-get install texlive-xetex texlive-fonts-recommended␣
 ↪texlive-generic-recommended
```

[14]:
```
!jupyter nbconvert --to PDF /content/drive/My\ Drive/parcial laura/laura.ipynb
```

```
[NbConvertApp] WARNING | pattern u'/content/drive/My Drive/parcial' matched no
files
[NbConvertApp] WARNING | pattern u'laura/laura.ipynb' matched no files
This application is used to convert notebook files (*.ipynb) to various other
formats.

WARNING: THE COMMANDLINE INTERFACE MAY CHANGE IN FUTURE RELEASES.

Options
-------

Arguments that take values are actually convenience aliases to full
Configurables, whose aliases are listed on the help line. For more information
on full configurables, see '--help-all'.

--execute
    Execute the notebook prior to export.
--allow-errors
    Continue notebook execution even if one of the cells throws an error and
include the error message in the cell output (the default behaviour is to abort
conversion). This flag is only relevant if '--execute' was specified, too.
--no-input
    Exclude input cells and output prompts from converted document.
    This mode is ideal for generating code-free reports.
--stdout
    Write notebook output to stdout instead of files.
--stdin
    read a single notebook file from stdin. Write the resulting notebook with
default basename 'notebook.*'
--inplace
    Run nbconvert in place, overwriting the existing notebook (only
    relevant when converting to notebook format)
-y
    Answer yes to any questions instead of prompting.
--clear-output
    Clear output of current file and save in place,
    overwriting the existing notebook.
--debug
    set log level to logging.DEBUG (maximize logging output)
--no-prompt
    Exclude input and output prompts from converted document.
--generate-config
    generate default config file
--nbformat=<Enum> (NotebookExporter.nbformat_version)
    Default: 4
```

```
    Choices: [1, 2, 3, 4]
    The nbformat version to write. Use this to downgrade notebooks.
--output-dir=<Unicode> (FilesWriter.build_directory)
    Default: ''
    Directory to write output(s) to. Defaults to output to the directory of each
    notebook. To recover previous default behaviour (outputting to the current
    working directory) use . as the flag value.
--writer=<DottedObjectName> (NbConvertApp.writer_class)
    Default: 'FilesWriter'
    Writer class used to write the  results of the conversion
--log-level=<Enum> (Application.log_level)
    Default: 30
    Choices: (0, 10, 20, 30, 40, 50, 'DEBUG', 'INFO', 'WARN', 'ERROR',
'CRITICAL')
    Set the log level by value or name.
--reveal-prefix=<Unicode> (SlidesExporter.reveal_url_prefix)
    Default: u''
    The URL prefix for reveal.js (version 3.x). This defaults to the reveal CDN,
    but can be any url pointing to a copy  of reveal.js.
    For speaker notes to work, this must be a relative path to a local  copy of
    reveal.js: e.g., "reveal.js".
    If a relative path is given, it must be a subdirectory of the current
    directory (from which the server is run).
    See the usage documentation
    (https://nbconvert.readthedocs.io/en/latest/usage.html#reveal-js-html-
    slideshow) for more details.
--to=<Unicode> (NbConvertApp.export_format)
    Default: 'html'
    The export format to be used, either one of the built-in formats
    ['asciidoc', 'custom', 'html', 'latex', 'markdown', 'notebook', 'pdf',
    'python', 'rst', 'script', 'slides'] or a dotted object name that represents
    the import path for an `Exporter` class
--template=<Unicode> (TemplateExporter.template_file)
    Default: u''
    Name of the template file to use
--output=<Unicode> (NbConvertApp.output_base)
    Default: ''
    overwrite base name use for output files. can only be used when converting
    one notebook at a time.
--post=<DottedOrNone> (NbConvertApp.postprocessor_class)
    Default: u''
    PostProcessor class used to write the results of the conversion
--config=<Unicode> (JupyterApp.config_file)
    Default: u''
    Full path of a config file.

To see all available configurables, use `--help-all`
```

```
Examples
--------

    The simplest way to use nbconvert is

    > jupyter nbconvert mynotebook.ipynb

    which will convert mynotebook.ipynb to the default format (probably HTML).

    You can specify the export format with `--to`.
    Options include ['asciidoc', 'custom', 'html', 'latex', 'markdown',
'notebook', 'pdf', 'python', 'rst', 'script', 'slides'].

    > jupyter nbconvert --to latex mynotebook.ipynb

    Both HTML and LaTeX support multiple output templates. LaTeX includes
    'base', 'article' and 'report'.  HTML includes 'basic' and 'full'. You
    can specify the flavor of the format used.

    > jupyter nbconvert --to html --template basic mynotebook.ipynb

    You can also pipe the output to stdout, rather than a file

    > jupyter nbconvert mynotebook.ipynb --stdout

    PDF is generated via latex

    > jupyter nbconvert mynotebook.ipynb --to pdf

    You can get (and serve) a Reveal.js-powered slideshow

    > jupyter nbconvert myslides.ipynb --to slides --post serve

    Multiple notebooks can be given at the command line in a couple of
    different ways:

    > jupyter nbconvert notebook*.ipynb
    > jupyter nbconvert notebook1.ipynb notebook2.ipynb

    or you can specify the notebooks list in a config file, containing::

        c.NbConvertApp.notebooks = ["my_notebook.ipynb"]

    > jupyter nbconvert --config mycfg.py
```