Emacs + org-mode + python in reproducible research

John Kitchin

Department of Chemical Engineering,
Carnegie Mellon University
Pittsburgh, PA

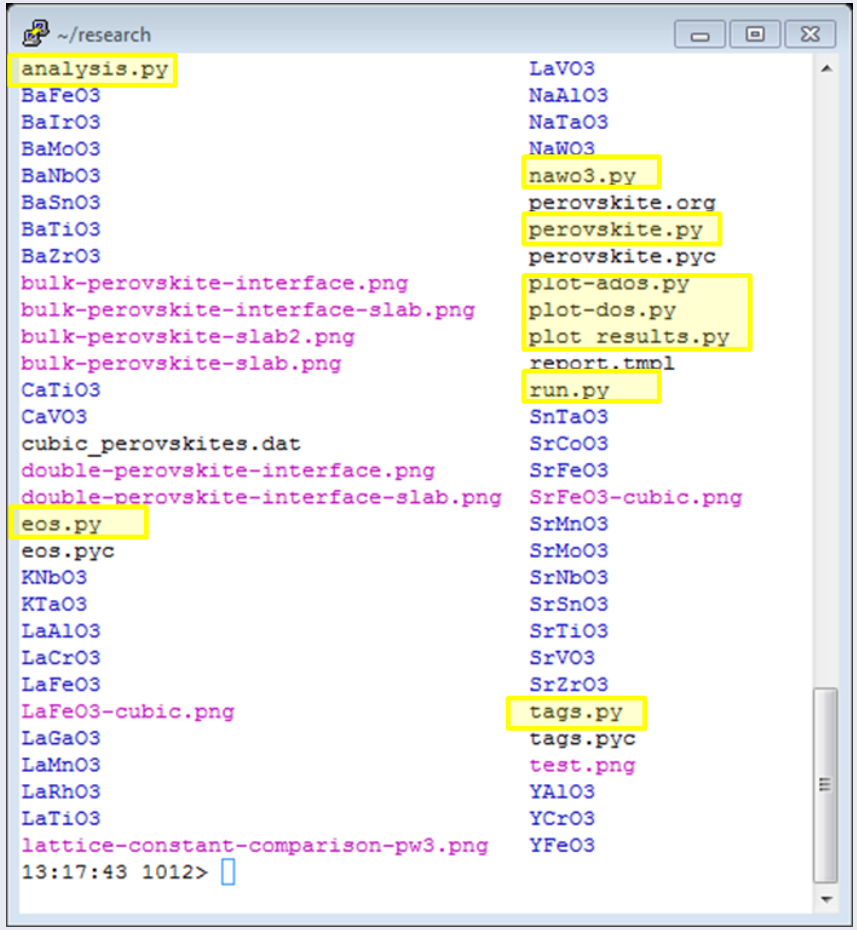
2013-06-27 Thu

Problem to solve #1

Computational research workflow

1. Setup a lot of calculations (perovskite.py)
2. Run calculations (run.py) a. Fix a few problem calculations by hand (nawo3.py
3. Analyze calculations (analysis.py, plot*.py,...)
4. ...(scripts4-8, miscellaneous command line work)
5. Try to teach student steps 1-4
6. Or, try to repeat steps 1-4 myself...

Lots of scripts



Problem to solve #2

- Integrating derivation of methods with illustrative code examples
  - Writing math in comments is tedious
  - Pasting code and results into text is tedious
  - Tedious = error-prone
  - Or in my case: not likely to happen

Now, we consider the integral to compute the electronic entropy. The entropy is proportional to this integral.

$$\int n(\epsilon)(f \ln f + (1-f)\ln(1-f))d\epsilon$$

It looks straightforward to compute, but it turns out there is a wrinkle. Evaluating the integrand leads to `nan` elements because the ln(0) is -∞.

```
import numpy as np
mu = 0
k = 8.6e-5
T = 100

def fermi(e):
    return 1.0 / (np.exp((e - mu)/(k*T)) + 1)

espan = np.array([-20, -10, -5, 0.0, 5, 10])
f = fermi(espan)

print f * np.log(f)
print (1 - f) * np.log(1 - f)
```

```
[  0.00000000e+000   0.00000000e+000   0.00000000e+000
-3.46573590e-001
  -1.85216532e-250               nan]
[         nan          nan          nan -0.34657359  0.
0.         ]
```

In this case, these `nan` elements should be equal to zero (x ln(x) goes to zero as x goes to zero). So, we can just ignore those elements in the integral. Here is how to do that.

```
import numpy as np
import matplotlib.pyplot as plt

mu = 0
k = 8.6e-5
T = 1000
```

Problem to solve #3

The issue

- How did I make this figure?
  - Where is the script?
  - Where is the data?
  - How did I make the data?
- Or how do I include the data in this figure from another paper in my current paper?

Data and methods tend to get lost over time as students leave, old computers die, ...

A figure from a manuscript

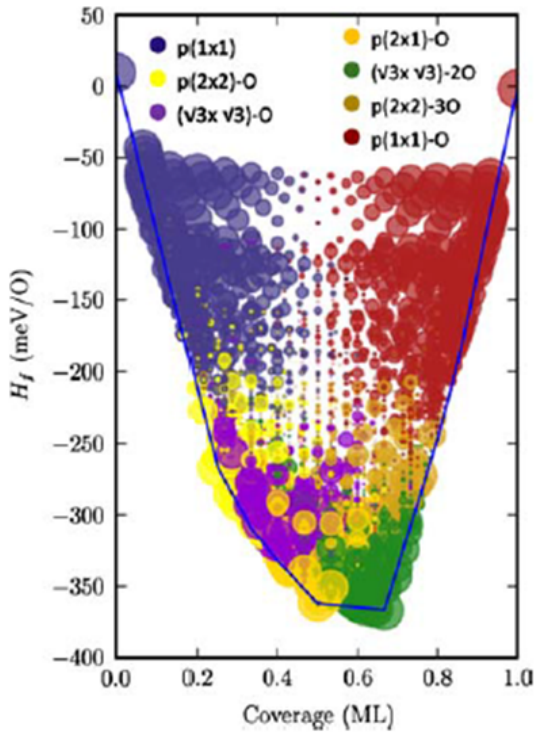


Figure 8. Similarity diagram for the platinum surface. Circles have been plotted for every configuration which show the configuration's similarity to the reference configurations at the top of the figure. The colour of the circle corresponds to the reference configuration, and the radius of the circle is proportional to the configurations similarity to the reference configuration.

Miller, Spencer D. and Kitchin, John R.(2009), *Uncertainty and figure selection for DFT based cluster expansions for oxygen adsorption on Au*

These problems have related solutions

- Problem 1 (documenting computational workflow)
  - Solved if we can do all that work in single document and keep a record of the results of each step
- Problem 2 (integrating mathematics with code)
  - Solved if text, data and code can be easily interspersed, and code can be run and output readily captured
- Problem 3 (how did I make the figure)
  - Solved if we can keep everything together and export what we want in the form we need
- The solution is an editor that can interact with the computer, a markup language that separates code, data and text, and a convenient programming language
- I will present the combination of these that works best for me:

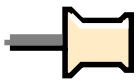
Emacs + org-mode + Python

Emacs in a nutshell

- Emacs is an extensible editor
  - Extensible in Emacs-Lisp, a full programming language
    - Users can customize every aspect of the editor
    - You can add any functionality you want
    - Like a "browser" for text
  - Operates in "modes" that provide features
    - Every major language has a mode: Python, C/C++, Fortran, Shell, LaTeX, markdown, restructured Text, etc...
    - provides editing functions, syntax highlighting, etc...
  - Provides complete integration with the operating system
    - This enables system commands to be run, and the output captured

Org-mode (http://orgmode.org/)

"Org mode is for keeping notes, maintaining TODO lists, planning projects, and authoring documents with a fast and effective plain-text system."

- Org-mode is written in Emacs-Lisp
- Outline mode that enables document organization
- Amazing task management capability
- Lightweight markup language that differentiates text, data and code
- You can embed arbitrary LaTeX, HTML, tables, etc... in it
- Code is executable in the editor, and the results are captured in the editor
- Enables navigatable "links" to files, commands, locations, urls, ...
- Export engine that converts selected content to PDF, LaTeX, HTML, ascii, etc... (e.g. this presentation!) 

Example - shell scripts

```
ls | sort
```

```
archive
blog.png
dft-book-1.png
fe-ni-al.png
fig8.png
header.png
kitchin-emacs-orgmode-python.org
kitchin-emacs-orgmode-python.pdf
kitchin-emacs-orgmode-python.tex
ls.png
pycse-1.png
pycse-2.png
```

Example with python code

```
import os
files =  os.listdir('.')
files.sort()
for f in files: print f
```

```
archive
blog.png
dft-book-1.png
fe-ni-al.png
fig8.png
header.png
kitchin-emacs-orgmode-python.org
kitchin-emacs-orgmode-python.pdf
kitchin-emacs-orgmode-python.tex
ls.png
pycse-1.png
pycse-2.png
```

Example with emacs-lisp

```
(mapcar (lambda (arg)
  (princ (format "%s\n" arg)))
(directory-files "."))
```

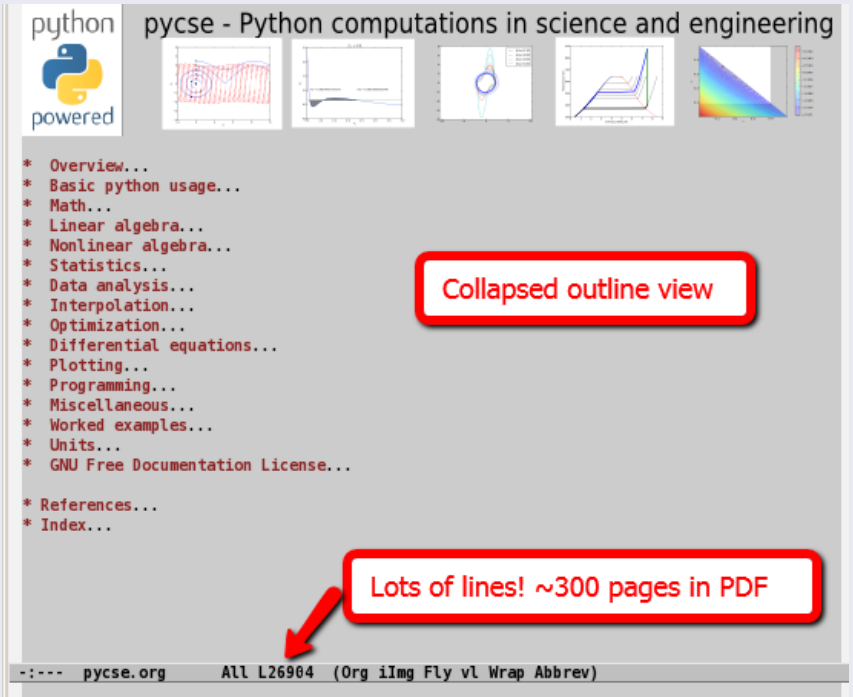
```
.
..
archive
blog.png
dft-book-1.png
fe-ni-al.png
fig8.png
header.png
kitchin-emacs-orgmode-python.org
kitchin-emacs-orgmode-python.pdf
kitchin-emacs-orgmode-python.tex
ls.png
pycse-1.png
```

Emacs + org-mode projects

- PYCSE - `http://jkitchin.github.io/pycse`
  - E-book on python calculations in science and engineering (~300 pages)
- Python blog - `http://jkitchin.github.io`
  - 169 posts on mostly python, created and published using org-mode and blogofile
- dft-book - `http://jkitchin.github.io/dft-book`
  - E-book on using python to drive quantum chemistry to compute material properties (~300 pages)
- Two scientific manuscripts submitted
  - "Simulating temperature programmed desorption of oxygen on Pt(111) using DFT derived coverage dependent desorption barriers" to Topics in Catalysis
  - "Effects of O_2 and SO_2 on the capture capacity of a primary-amine based polymeric CO_2 sorbent" to Industrial & Engineering Chemistry Research
  - Manuscripts and supporting information were generated in Emacs + org-mode, and exported to LaTeX for submission

PYCSE - http://jkitchin.github.io/pycse

Document overview



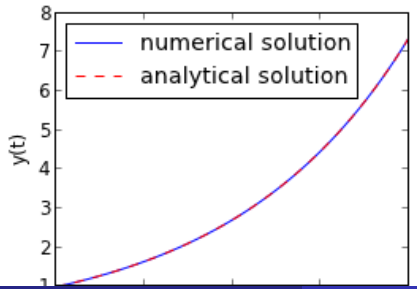
- Code is written and executed in the editor. Output captured.
- Exported to blog, HTML and PDF. Mobi and ePub are also possible.

A subsection of the document

```
*** Numerical solution to a simple ode
    :PROPERTIES:
    :categories: ODE, interpolation
    :date:       2013/02/26 21:17:44
    :updated:    2013/03/23 16:03:44
    :END:
Matlab post

Integrate this ordinary differential equation (ode):

$\frac{dy}{dt} = y(t)$

over the time span of 0 to 2. The initial condition is y(0) = 1.

to solve this equation, you need to create a function of the form: dydt = f(y, t) and
then use one of the odesolvers, e.g. odeint.

#+BEGIN_SRC python
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt

def fprime(y,t):
    return y

tspan = np.linspace(0, 2)
y0 = 1
ysol = odeint(fprime, y0, tspan)
plt.figure(figsize=(4,3))
plt.plot(tspan, ysol, label='numerical solution')
plt.plot(tspan, np.exp(tspan), 'r--', label='analytical solution')
plt.xlabel('time')
plt.ylabel('y(t)')
plt.legend(loc='best')
plt.savefig('images/simple-ode.png')
#+END_SRC

#+RESULTS:
```

information about blog post

inline, rendered math

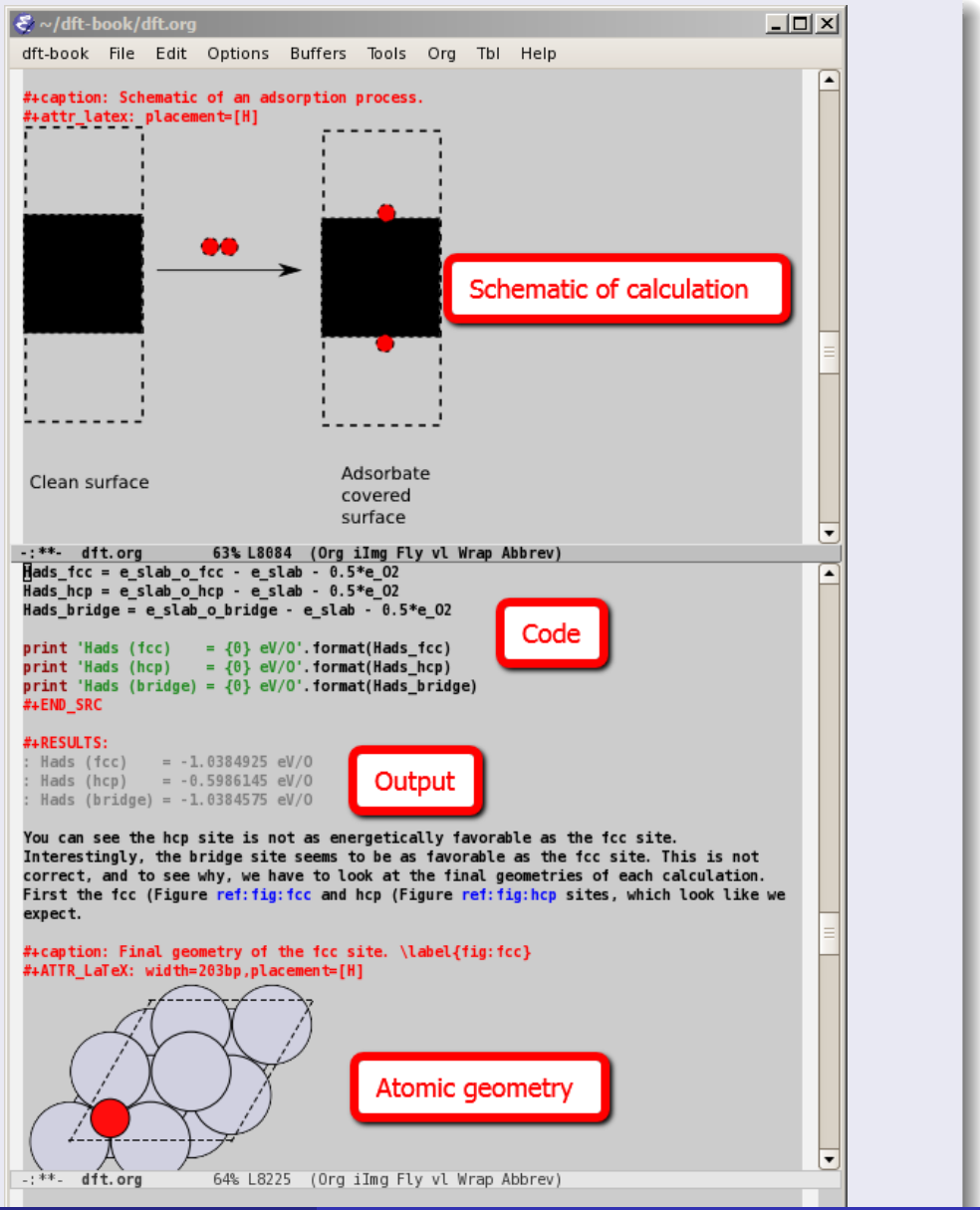


inline figure

dft-book - http://jkitchin.github.io/dft-book

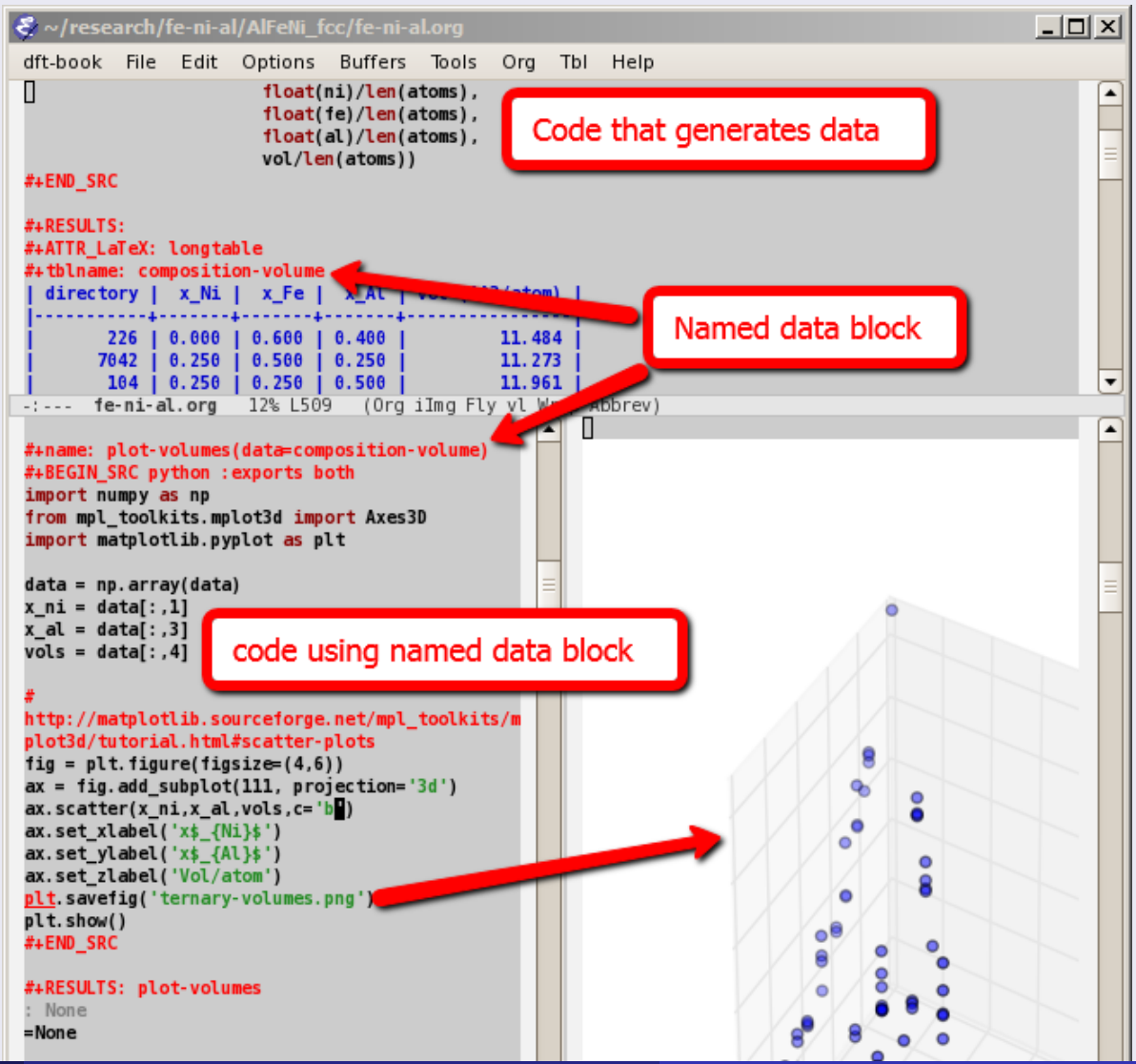
Embedded text, math, code and output.

- 300+ pages of using python to run quantum chemical calculations
- might be 50+% code!
- Every example written and run in the book
  - no cut and paste code/results
  - It ran correctly *at least once*



Org-mode in documenting computational/research workflow

- Separation of data generation and analysis promotes data reuse
- Easier to read scripts



Challenges

- Org-mode is deeply integrated with Emacs
  - pro - You get all the power of Emacs
  - on the other hand - You have to learn Emacs and Emacs-Lisp
  - Other editors can mimic the capabilities
- Org-mode is markup *and* functionality
  - restructured text + Sphinx is the closest in spirit
  - has extensibility (in Python!)
  - currently lacks editor integration even in Emacs
- Getting exported format perfect can be challenging
  - this is a general problem with converting formats

I wish we had an editor as powerful as Emacs that is extensible in Python.

Conclusions

- Reproducible research needs new tools, new workflows
  - Users will probably need to customize tools for their needs
- Emacs + org-mode was a game changer in reproducible research for me. It enabled
  - Authoring two books on using python in science and engineering
  - A python based blog
  - Documenting computational work
  - Managing the work-life of an engineering professor
- The key features that enabled this are
  - Extensible editor
  - Extensible markup language
  - Scripting (Python + others)

Thanks for your attention!

`https://github.com/jkitchin/scipy2013`