数式処理 *Bulletin of JSSAC (2019)*
Vol. XX, No. XX, pp. 1 - 2

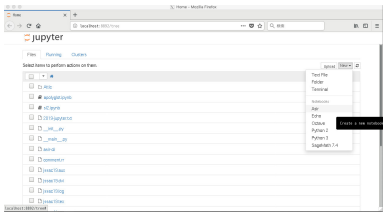
大会報告

Jupyter notebook への asir カーネルの実装 (実装報告)

高山信毅 *

神戸大学大学院理学研究科数学専攻

Jupyter[1] Notebook は python その他さまざまな言語のためのノートブックインタフェースである. Jupyter とカーネルは http でブラウザと通信するサーバー. Notebook はブラウザで動く.



ノートブックは Cell で構成されており, プログラム以外に Markdown により文書も記述できる. TEX による数式は MathJax により表示される. shift + enter (または play ボタン) で整形. Double click で markdown 編集モードになる. 次の図は Markdown の例.

```
# 因数分解
$x^2-1$ の因数分解を asir でやってみた.
 - [f,n] は $f^n$ を表す.
  - $f$ は ${\bf Q}[x]$ で既約.
 - [1,1] の部分は定数係数.
コードは sharp tab
# tab
    fctr(x^3-1);
バッククオートで文章中にコードを書ける `fctr(x^2-1);`
```

因数分解

$x^3 - 1$ の因数分解を asir でやってみた.

- [f,n] は f^n を表す.
  - f は $\mathbf{Q}[x]$ で既約.
- [1,1] の部分は定数係数.数字で箇条書き
- [f,n] は f^n を表す.
- [1,1] の部分は定数係数. コードは4つの空白.

```
fctr(x^3-1);
```

バッククオートで文章中のコード `fctr(x^2-1);`

またプログラムには行番号を表示すると便利な場合が多いが, これは esc l で可能.

*takayama@math.kobe-u.ac.jp

1) https://juyter.org

我々は asir を Jupyter notebook からも使えるようにした. 実装方針は以下のとおり.

1. python で書かれた octave kernel を再利用.
2. python から asir の接続には, OpenXM[2] の sm1 toolkit を使う. toolkit を shared library にして python と接続する方法と, 標準入出力で python と接続する方法 (`pexpect module` などを活用) があるが, ここでは octave kernel を再利用する方法を用いるので, toolkit を中間層として asir に octave 風の動作をさせることで (実装では `ox_texmacs -view jupyter` で) 標準入出力を用いて接続する.

OpenXM で通信プロトコルが厳密に定義されており, また C 言語による toolkit が用意されているので, プロトタイプは 6 時間ほどで完成した.

kernel の作成方法や基本的な仕組みについては Making kernel for Jupyter[3] に解説や例がある. 発表のスライドファイル [4] にこの記事の日本語要約を掲載しているので興味のある人は参考にしてもらいたい.

覚書として Jupyter asir kernel の実装時に注意したこと等を記載しておく.

1. `base_prompt()` で Prompt を変更するふり. IPython カーネルは pexpect を使ってる.
2. code は `do_execute_direct` で実行される.
3. make_figures で return None とすれば'NoneType' object has no attribute '__getitem__' エラーが例外でおきるのを抑制.
4. asir_kernel/ にも __init__.py 必要.
5. `ctrl("debug_window",0); ctrl("no_debug_on_error",1);` (`jupyter-init.rr`)
6. 改行で prompt を戻すが buffering するのみ. ';;' で計算開始とした.
7. Debug について. `ox_texmacs.c` で `DEBUG2` を def して `/tmp/debug-texmacs.txt`. サーバーのログを表示するには `export OX_XTERM_GEOMETRY=80x20+0+0` してスタート. `kernel.py` では

```
stream_handler = None if silent else self.stream_handler
if self.logger:
    self.logger.setLevel(logging.DEBUG) <-- これを加えた.
    self.logger.debug('Asir eval:')
    self.logger.debug(code)
f = open('tmptmp.txt','a');f.write(str(code));f.close()
self._octave_engine.logger.debug(str(code)) <-- これを加えた.
val = ProcessMetaKernel.do_execute_direct(self, code, silent=silent)
```

8. `http://pygments.org` (get_pygments_lexer) で入力の構文を検査. asir_kernel/kernel.py, `language_info` で C を指定.
9. jupyter から呼ぶ shell script は `#!/bin/bash` が必要. さもなければ Exec format error.
10. Debian では `~./local` の下に pip 等でインストールしたのものが存在. たとえば `~/.local/bin/jupyter-kernelspec list` で使える kernel の一覧を得る.

なお参考文献は本文中に記載した URL を参照してほしい.

[2] `http://www.openxm.org`
[3] `https://jupyter-client.readthedocs.io/en/stable/kernels.html`
[4] `http://www.math.kobe-u.ac.jp/HOME/taka/2019/20190601-jupyter.pdf`