やさしい Python/pySALEPLotの つかいかた …

わきた しげる

Python 1/5

- Pythonとは？
  - 動的プログラミング言語（コンパイル不要）
- なぜPython？
  - 可読性・メンテナンス性に優れている
  - すぐに書ける
  - Numpyなどのmodule(package)が抱負

Python 2/5

- Numpy
  - 配列や行列の基本タイプ、
    それらの数値計算のためのmodule
- Matplotlib
  - グラフ描画のためのmodule
- iPython
  - 対話型インターフェイスのシェル
    - タブ補完が便利
    - （ipython --matplotlib での起動も便利）

Python 3/5

- ここからしばらくは解析サーバにて
  - ssh -Y isale??@an.cfca.nao.ac.jp
- ログイン後
  - module load anaconda/2 intel/14.0
- iPython 起動
  - pySALEPlot.py がある場所に移動してから
    - cd iSALE-Dellen
    - ipython

Python 3/5

- PythonとiPythonでversionを確認しよう

```
[wakitasg(33715)@an08:~/161109isale]$python --version
Python 2.7.11 :: Anaconda 2.5.0 (64-bit)
[wakitasg(33716)@an08:~/161109isale]$cat version.py
import sys
print sys.version
[wakitasg(33717)@an08:~/161109isale]$python version.py
2.7.11 |Anaconda 2.5.0 (64-bit)| (default, Dec  6 2015, 18:08:32)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-1)]
[wakitasg(33718)@an08:~/161109isale]$ipython --version
4.0.3
[wakitasg(33719)@an08:~/161109isale]$ipython
Python 2.7.11 |Anaconda 2.5.0 (64-bit)| (default, Dec  6 2015, 18:08:32)
Type "copyright", "credits" or "license" for more information.

IPython 4.0.3 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: import sys

In [2]: print sys.version
2.7.11 |Anaconda 2.5.0 (64-bit)| (default, Dec  6 2015, 18:08:32)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-1)]

In [3]:
```

Python 4/5

- import *** で、必要なmoduleを読み込む

```
[wakitasg(33715)@an08:~/161109isale]$python --version
Python 2.7.11 :: Anaconda 2.5.0 (64-bit)
[wakitasg(33716)@an08:~/161109isale]$cat version.py
import sys
print sys.version
[wakitasg(33717)@an08:~/161109isale]$python version.py
2.7.11 |Anaconda 2.5.0 (64-bit)| (default, Dec  6 2015, 18:08:32)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-1)]
[wakitasg(33718)@an08:~/161109isale]$ipython --version
4.0.3
[wakitasg(33719)@an08:~/161109isale]$ipython
Python 2.7.11 |Anaconda 2.5.0 (64-bit)| (default, Dec  6 2015, 18:08:32)
Type "copyright", "credits" or "license" for more information.

IPython 4.0.3 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: import sys

In [2]: print sys.version
2.7.11 |Anaconda 2.5.0 (64-bit)| (default, Dec  6 2015, 18:08:32)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-1)]

In [3]:
```

Python 5/5

- print *** で、変数を出力する

```
[wakitasg(33715)@an08:~/161109isale]$python --version
Python 2.7.11 :: Anaconda 2.5.0 (64-bit)
[wakitasg(33716)@an08:~/161109isale]$cat version.py
import sys
print sys.version
[wakitasg(33717)@an08:~/161109isale]$python version.py
2.7.11 |Anaconda 2.5.0 (64-bit)| (default, Dec  6 2015, 18:08:32)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-1)]
[wakitasg(33718)@an08:~/161109isale]$ipython --version
4.0.3
[wakitasg(33719)@an08:~/161109isale]$ipython
Python 2.7.11 |Anaconda 2.5.0 (64-bit)| (default, Dec  6 2015, 18:08:32)
Type "copyright", "credits" or "license" for more information.

IPython 4.0.3 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: import sys

In [2]: print sys.version
2.7.11 |Anaconda 2.5.0 (64-bit)| (default, Dec  6 2015, 18:08:32)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-1)]

In [3]:
```

pySALEPlot 1/5

- iSALEのデータ('jdata.dat')をプロットするためのツール
  - Python の module
- import pySALEPlot as psp
  - import *** as ### とすることで、
    以降は ### で *** を呼び出せる
- model = psp.opendatfile(‘demo2D/jdata.dat’)
  - model という変数に、jdata.datの中身を入れる

pySALEPlot 2/5

- ファイルを読み込んでみよう
  - import pySALEPlot as psp
  - model = psp.opendatfile(‘demo2D/jdata.dat’)

```
In [7]: import pySALEPlot as psp

In [8]: model=psp.opendatfile('jdata.dat')
Opened iSALE data file 'jdata.dat', with 121 time steps

In [9]: step=model.タブ[tab]
model.alemode          model.filepos          model.nmproj          model.objrad          model.skipStep        model.xext
model.cellVolumes      model.fmax             model.nmtarx          model.objvel          model.skipToInit      model.xhires
model.cinfo            model.fmin             model.nmtary          model.path            model.skipToStep      model.xx
model.closeFile        model.geometry         model.nmtarz          model.plotSetupInfo   model.surface         model.y
model.cppr             model.gravityAnomaly   model.nplots          model.plottype        model.surfaceProfile  model.yc
model.craterGrowth     model.inputDict        model.nsteps          model.position        model.tracerInfo      model.yext
model.dx               model.inputParams      model.nvar            model.quickPlot       model.tracer_num      model.yhires
model.dy               model.itracers         model.nx              model.readStep        model.tracer_numu     model.yy
model.fieldlist        model.laststep         model.nxny            model.readvariable    model.tru
model.fieldmax         model.laynum           model.nxp             model.refden          model.units
model.fieldmin         model.magic            model.ny              model.scale           model.verbose
model.fileid           model.modelInfo        model.nyp             model.scalelabel      model.x
model.filename         model.nmat             model.objnum          model.setScale        model.xc
```

pySALEPlot 3/5

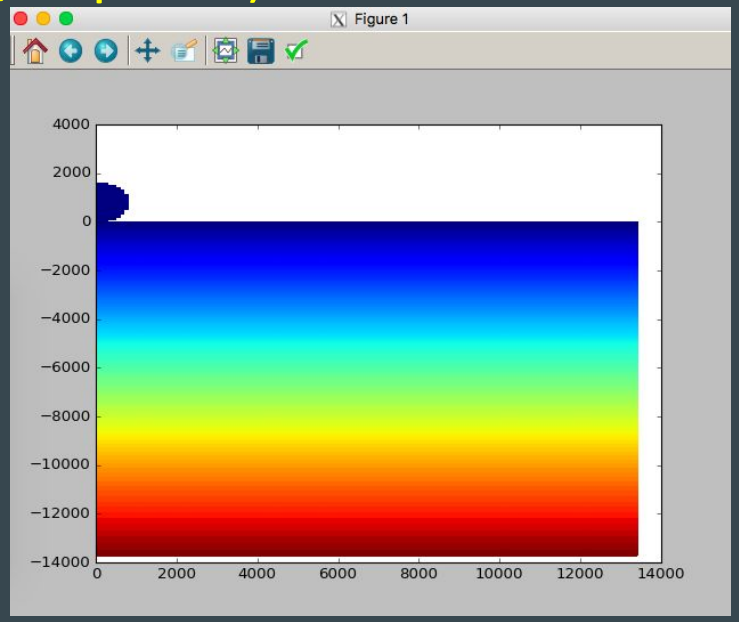
- 中身を少しだけみてみよう
  - step=model.readStep()
  - print step.Den

```
In [9]: step=model.readStep()
Read in ['Den'] for timestep -1 (0.000e+00 s)

In [10]: print step
<pySALEPlot.modelStep instance at 0x7f58ef5223f8>
In [12]: print step.タブ[tab]
step.Den        step.cyc        step.eof        step.filepos    step.mat        step.stepInfo   step.xmark
step.cmc        step.data       step.fileid     step.findTracer step.plottype   step.time       step.ymark

In [12]: print step.Den
[[3461.76611328125 3456.147216796875 3450.804443359375 ..., -- -- --]
 [3461.76611328125 3456.147216796875 3450.804443359375 ..., -- -- --]
 [3461.76611328125 3456.147216796875 3450.804443359375 ..., -- -- --]
 ...,
 [3461.76611328125 3456.147216796875 3450.804443359375 ..., -- -- --]
 [3461.76611328125 3456.147216796875 3450.804443359375 ..., -- -- --]
 [3461.76611328125 3456.147216796875 3450.804443359375 ..., -- -- --]]
```

pySALEPlot 4/5

- せっかくだからプロット
  - plt.pcolormesh(model.x,model.y,step.Den)
  - plt.show()

```
In [19]: plt.pcolormesh(model.x,model.y,step.Den)
Out[19]: <matplotlib.collections.QuadMesh at 0x7f58eb9075d0>
```



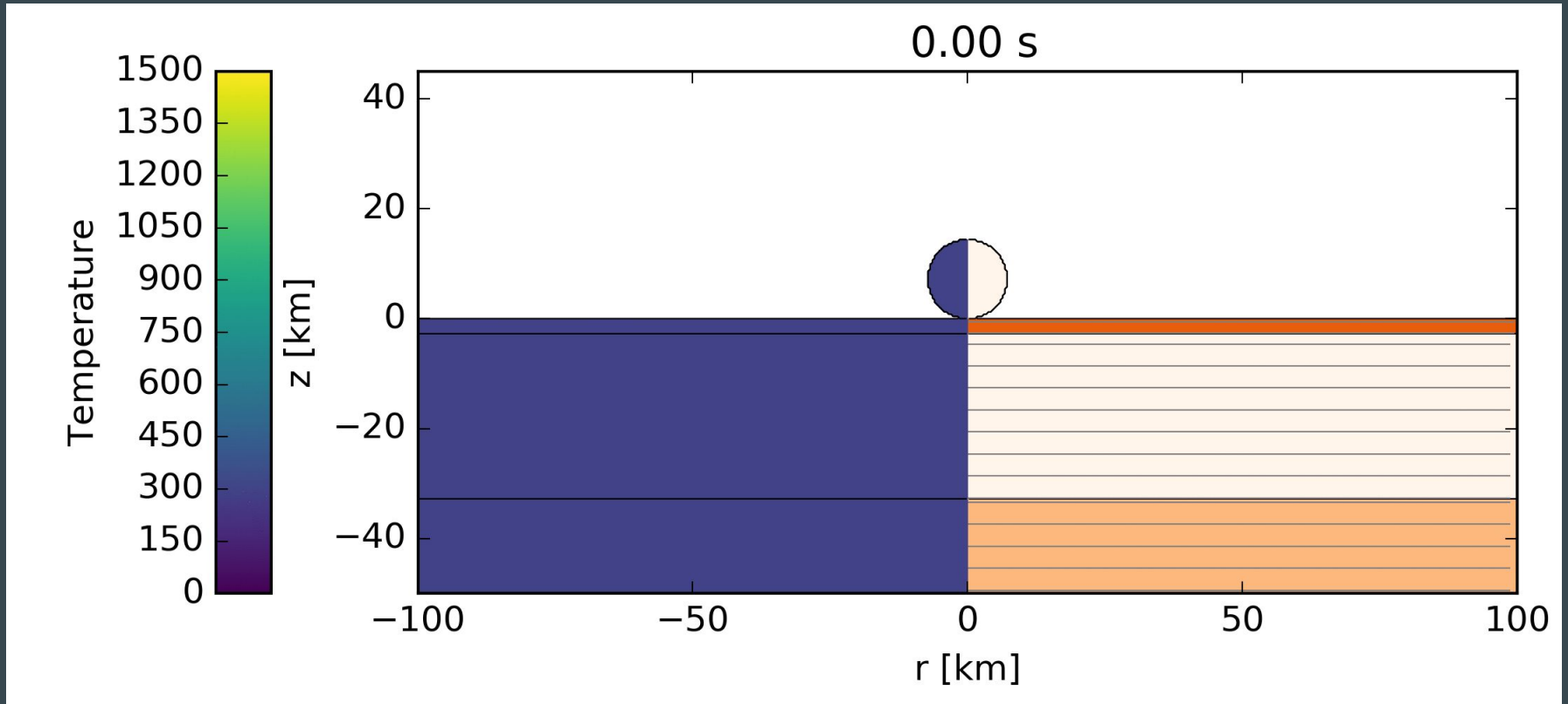
- plt とは？
  - matplotlibのこと
  - pySALEPlot.py の最初の方で呼び出している
    - import matplotlib.pyplot as plt

pySALEPlot 5/5

- 実際にプロットするときには、次を適当に組み合わせる
  - 時間
    - 動画を作るなら、繰り返し文(for)が必要
  - 変数
    - 密度[Den]、圧力[Pre]、温度[Tmp]、等々
  - プロットの種類
    - コンター（pcolormesh）
    - ライン（plot）
    - プロットの見栄えを整えるのは、一番最後
      - ラベル、カラーなど

ここまでのまとめ

- PythonやpySALEPlotってかんたんそう
- iSALEで計算したデータを簡単にpySALEPlotでプロットができそう



pySALEPLot 実践編

…

目標

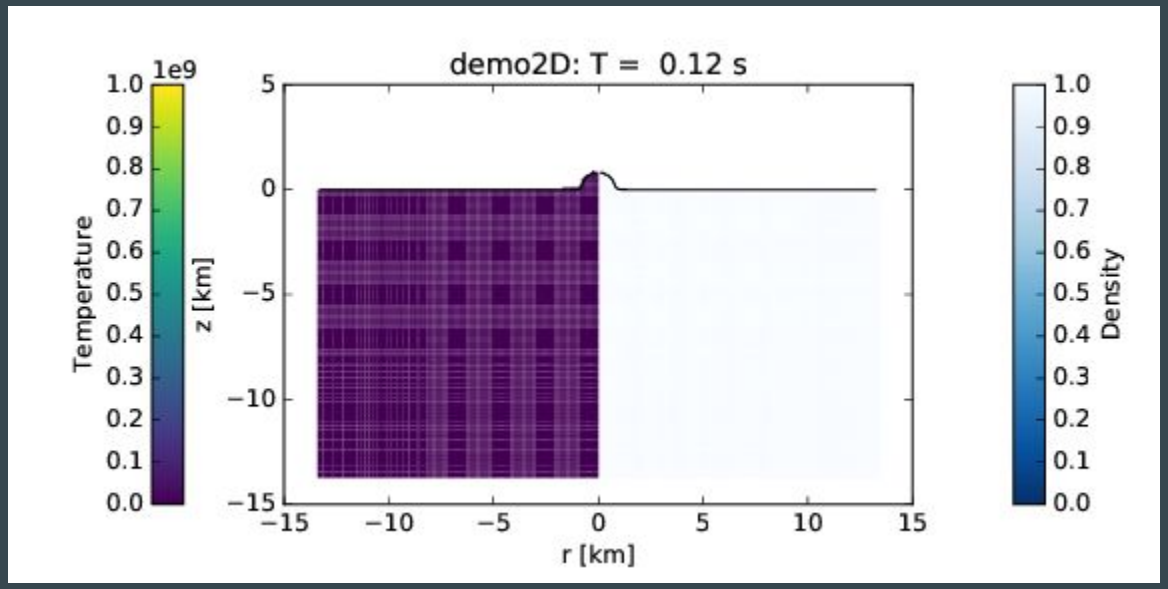
- pySALEPlotを用いた描画
  - かんたん編
    - pySALEPlot + python の仕組みを理解
  - 実践編
    - Sampleのpythonスクリプトを変更する
      - 出力したい変数に変える
      - トレーサー粒子を描き出す
      - 論文のために三角関数なども使ってみる

pySALEPlot: DenTmp.py

1. 読み込むデータの種類
   a. step=model.readStep(['Tmp', 'Den'],i)
2. 出力ファイル名変更
   a. fig.savefig('{}/DenTmp-{:05d}.eps'.format(dirname, i), format='eps', dpi=300)

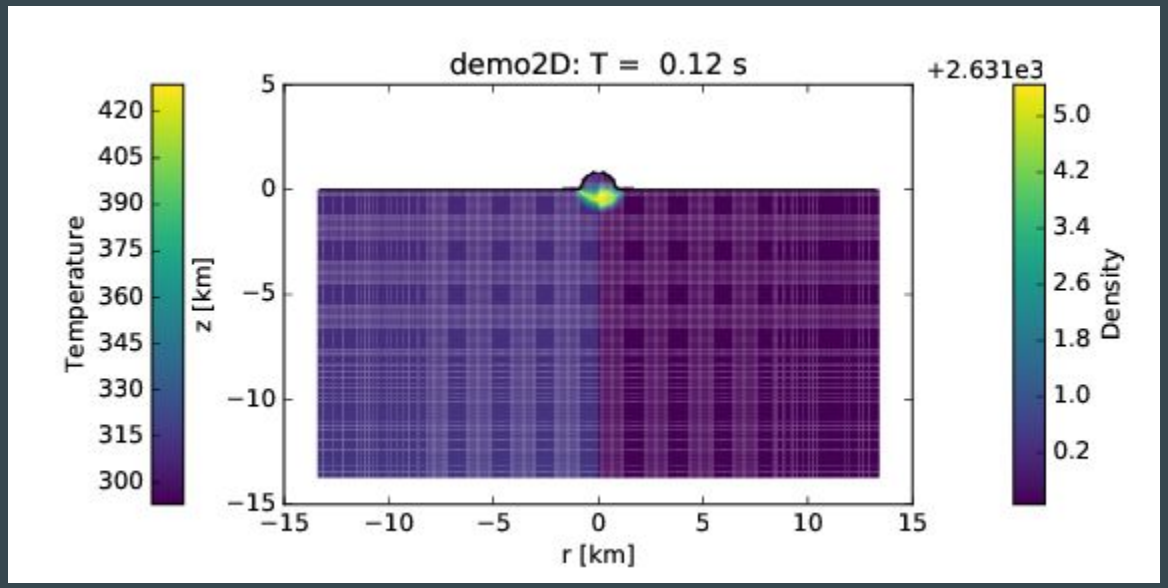
- python DenTmp.py

- plotされる値の範囲がおかしい



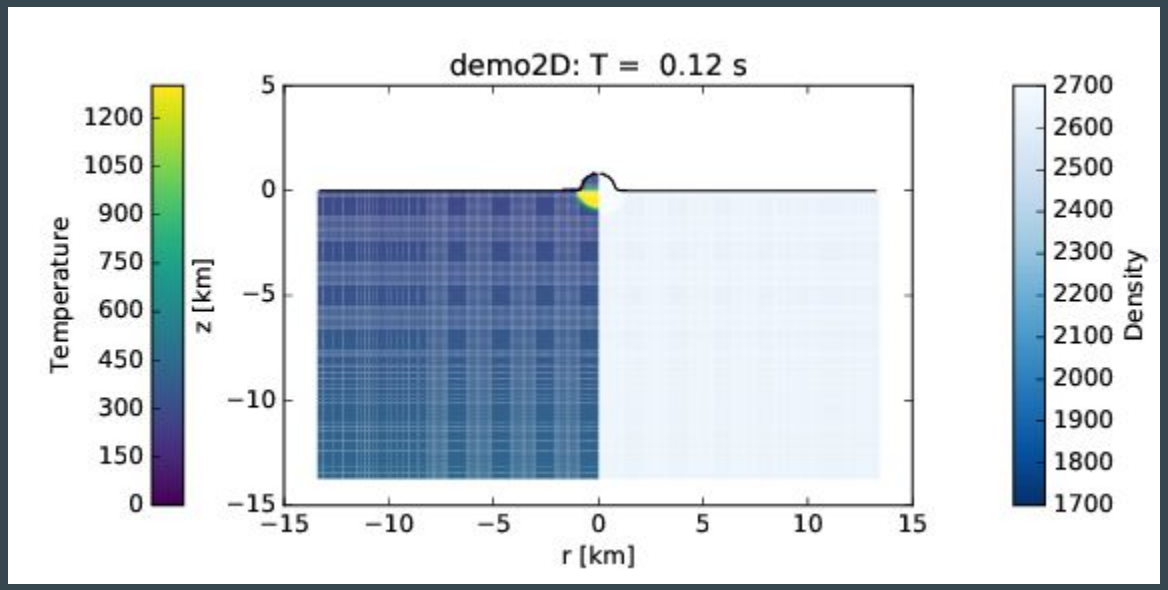
pySALEPlot: DenTmp.py

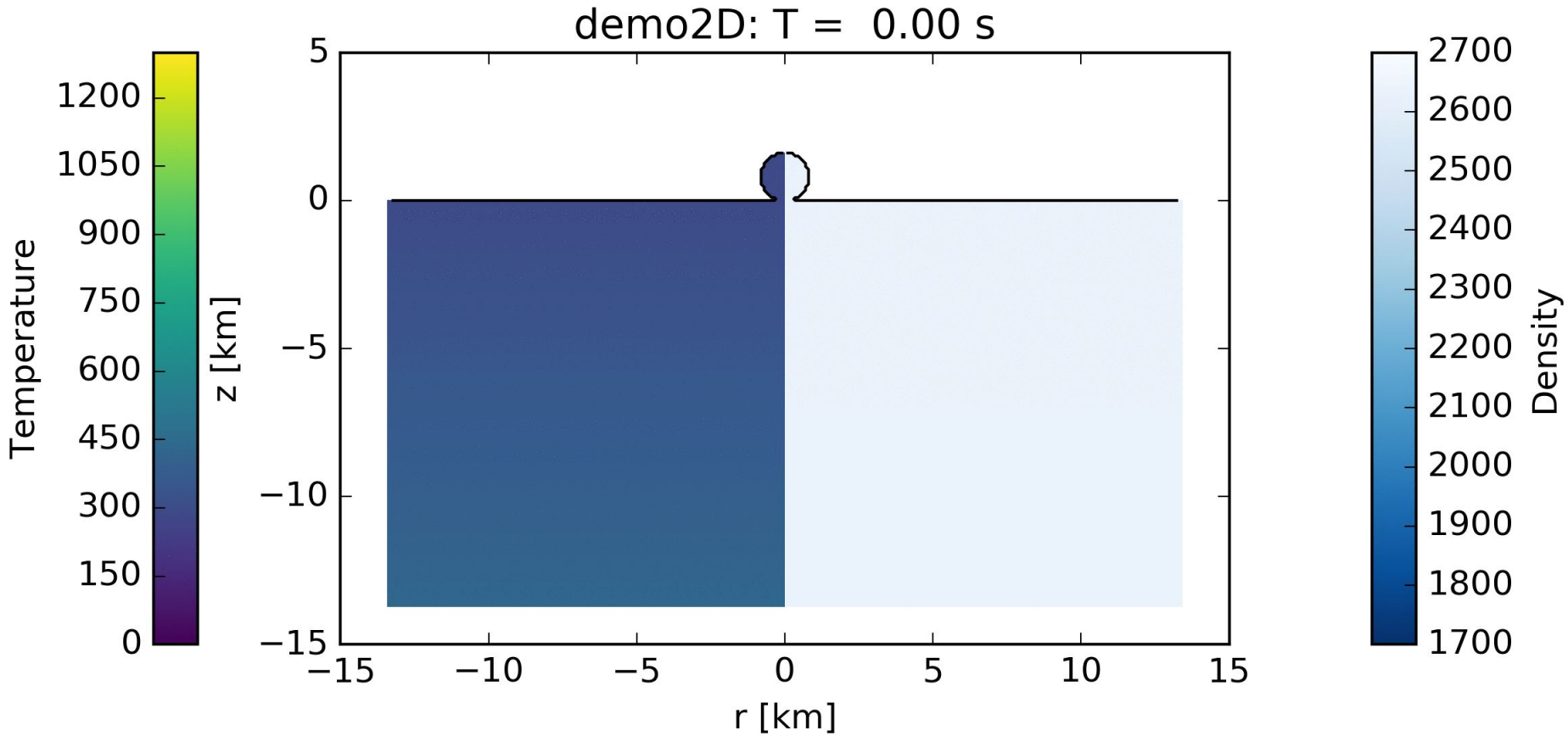
4. plotする値の範囲を変更
   a. p1=ax.pcolormesh(model.x, model.y, step.data[1])
   b. p2=ax.pcolormesh(-model.x, model.y, step.data[0])



pySALEPlot: DenTmp.py

4. plotする値の範囲を変更
   a. p1=ax.pcolormesh(model.x, model.y, step.data[1], cmap='Blues_r', vmin=1700., vmax=2700.)
   b. p2=ax.pcolormesh(-model.x, model.y, step.data[0], vmin=0., vmax=1300.)



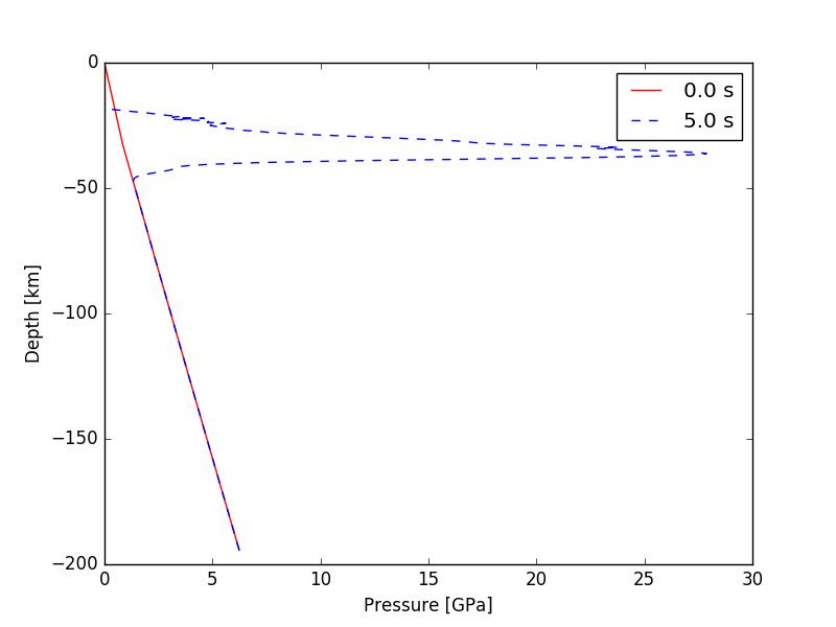

demo2D: T = 0.00 s
Temperature
1200
1050
900
750
600
450
300
150
0
z [km]
5
0
−5
−10
−15
−15
−10
−5
0
5
10
15
r [km]
Density
2700
2600
2500
2400
2300
2200
2100
2000
1900
1800
1700

pySALEPlot: DenTmp.py

5. 出力時間を元に戻す
   a. for i in np.arange(0,210,10):
6. 動画にする
   a. convert Plots/DenTmp*.eps DenTmp.gif
   b. ffmpeg -y -r 10 -i ./DenTmp/DenTmp-%04d0.png -s 800x400 DenTmp.mp4
      ただし、epsではなくpngで出力する必要がある
      mpl.use('Agg') #for png
      fig.savefig('{}/DenTmp-{:05d}.png'.format(dirname, i), format='png', dpi=300)

pySALEPlot: profile.py

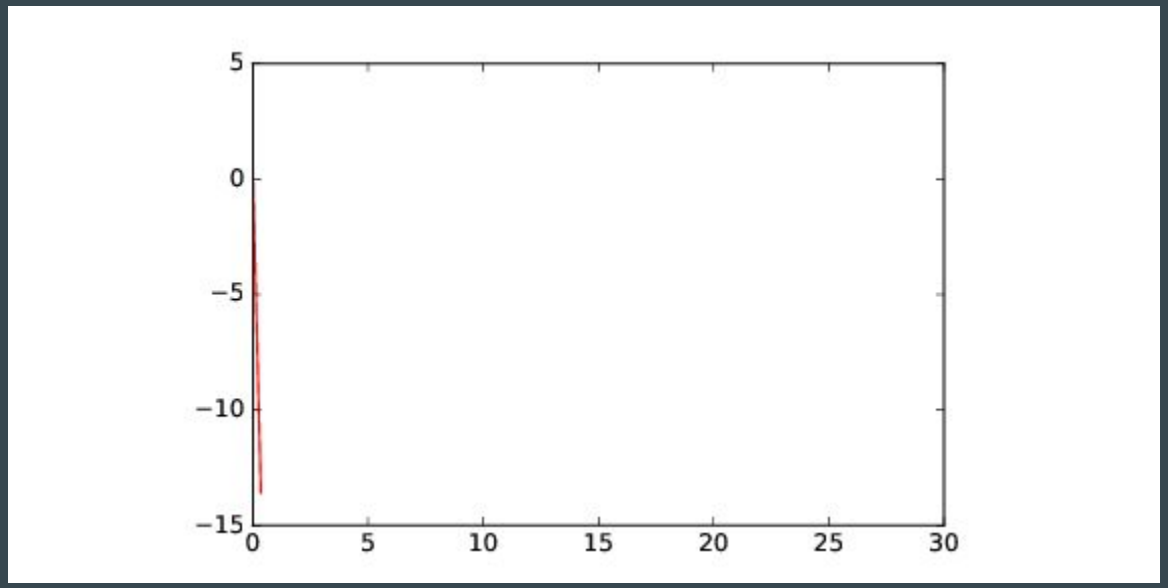
- x = 0(0.5) での 深さ方向(y 方向)の測線に沿った圧力プロファイル
- profile.py



pySALEPlot: profile.py

1. プロットしてみる
    a. step=model.readStep('Pre',0)
    b. ax.plot(step.Pre[0,:]*1e-9,model.yc[0,:],'r-')

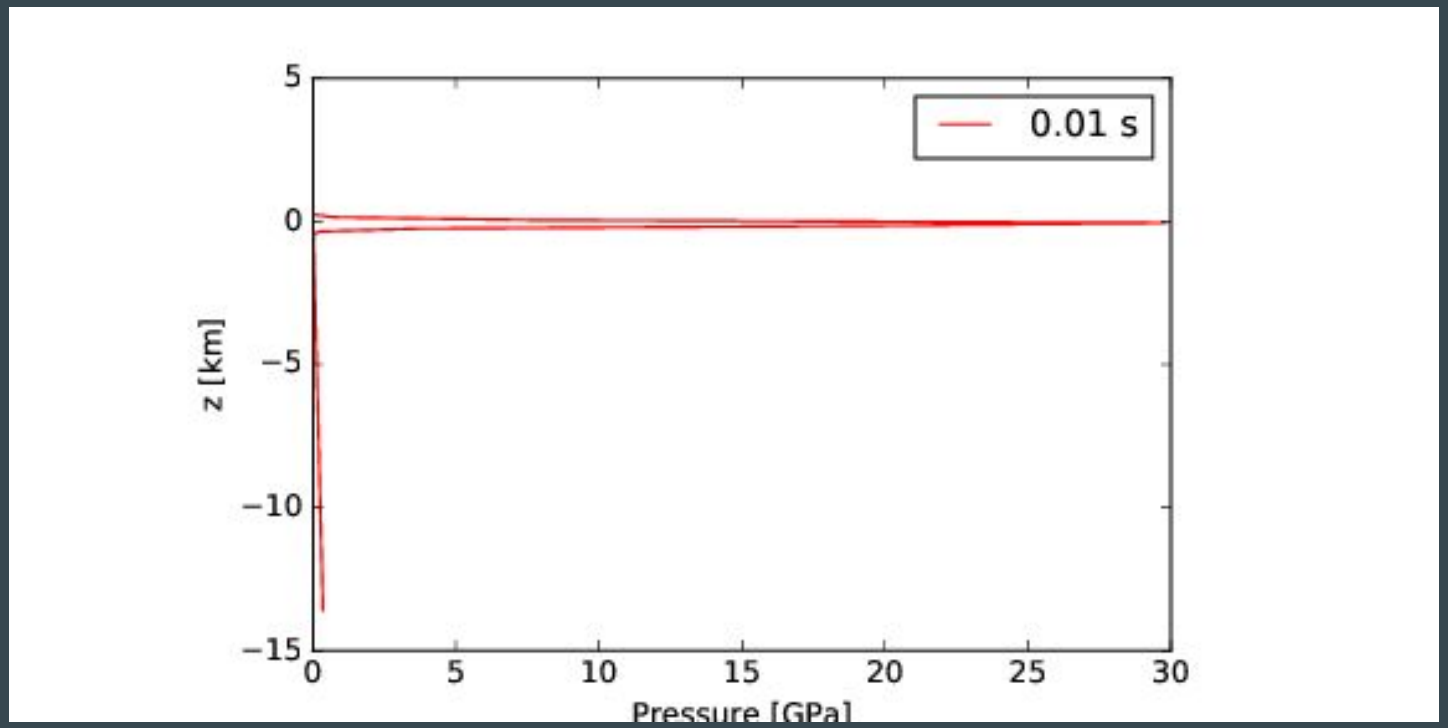
- 初期状態の圧力図



pySALEPlot: profile.py

2. プロットする時間の変更
   a. step=model.readStep('Pre',1)
3. 凡例をつけよう
   a. ax.plot(step.Pre[0,:]*1e-9,model.yc[0,:],'r-',label='{:4.2f}s'.format(step.time))
   b. ax.legend(loc=0)

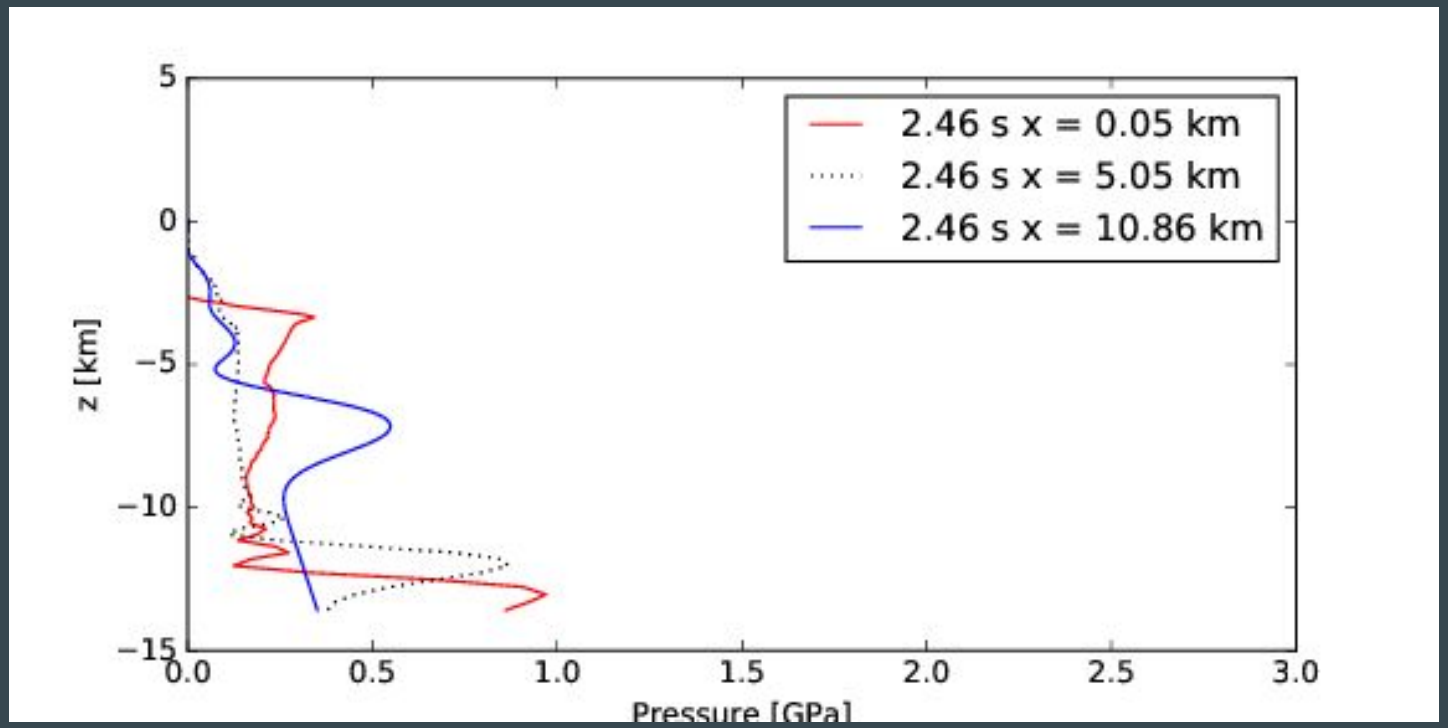
- 図らしくなった



pySALEPlot: profile.py

3. プロットする場所の変更
   a. ax.plot(step.Pre[0,:]*1e-9,model.yc[0,:])
      i. step.Pre と model.yc を理解する
      ii. len(step.Pre[0,:]) と len(model.yc[0,:]) は 140
          len(step.Pre[:,0]) と len(model.yc[:,0]) は 112
      iii. asteroid.inp の GRIDV/GRIDH (の合計) に対応
      iv. step.Pre[0,:] は model.xc[0,:] の場所でのPreに対応
      v. model.xc[0,:] は 全て 0.05 となっている

- 時間やプロットする場所を変えてみよう



pySALEPlot: profile.py

- step = model.readStep(‘Pre’,200)
- ax.plot(step.Pre[0,:]*1e-9,model.yc[0,:],'r-', label='{:4.2f} s x = {:4.2f} km'.format(step.time,model.xc[0,:][0]))
- ax.plot(step.Pre[50,:]*1e-9,model.yc[0,:],'k:', label='{:4.2f} s x = {:4.2f} km'.format(step.time,model.xc[50,:][0]))
- ax.plot(step.Pre[100,:]*1e-9,model.yc[0,:],'b-', label='{:4.2f} s x = {:4.2f} km'.format(step.time,model.xc[100,:][0]))

pySALEPlot: tracer.py

- トレーサー粒子の最大圧力をプロットする
- tracer.py

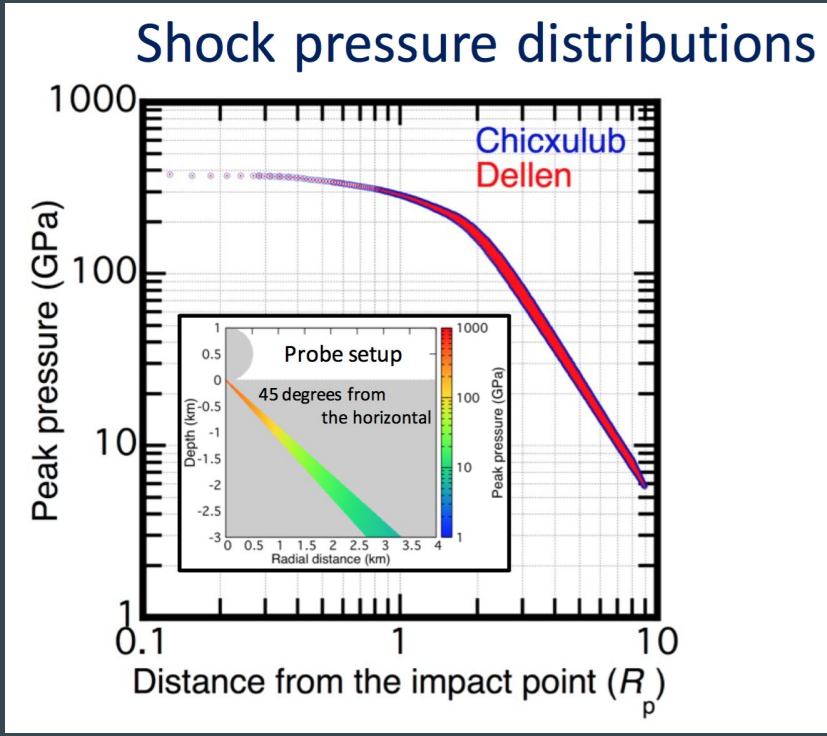
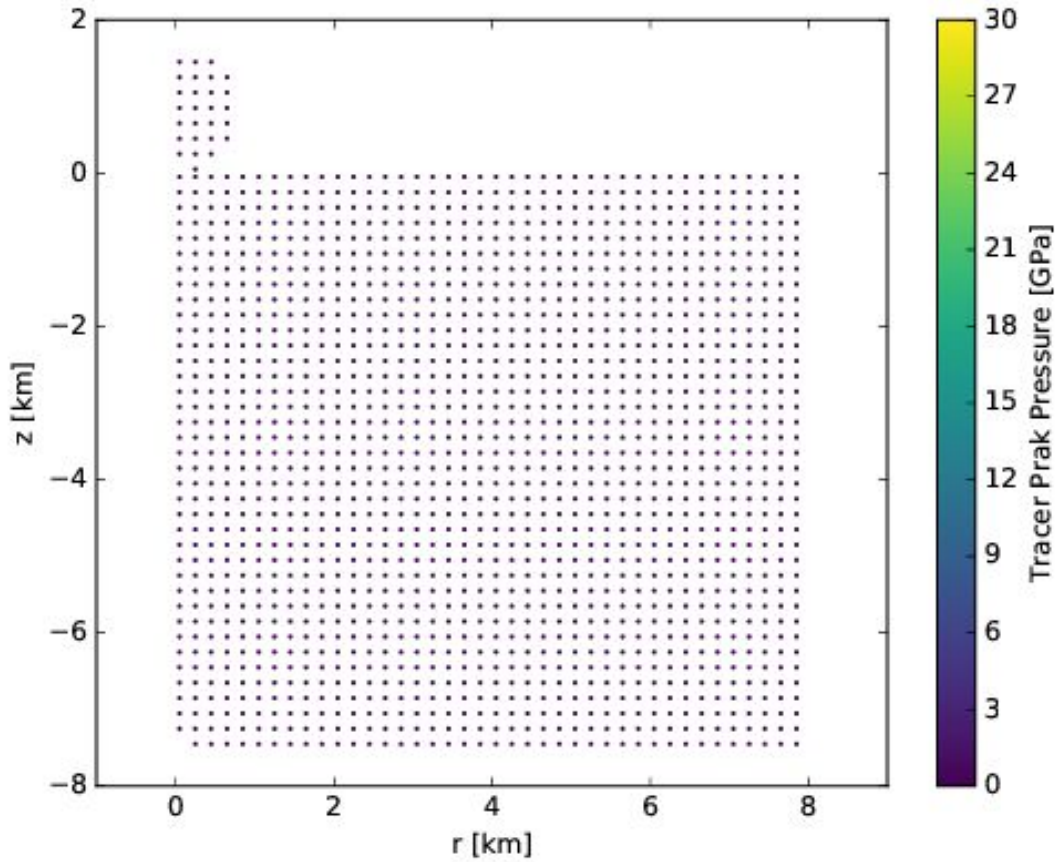


図:黒澤校長より寄贈

pySALEPlot: tracer.py

1. プロットしてみる
   a. step=model.readStep('TrP',0)
   b. scat = ax.scatter(step.xmark[tstart:tend],step.ymark[tstart:tend], c=step.TrP[tstart:tend]*1e-9,vmin=0,vmax=30, cmap='viridis',s=4,linewidths=0)


2
0
−2
−4
−6
−8
z [km]
0
2
4
6
8
r [km]
30
27
24
21
18
15
12
9
6
3
0
Tracer Prak Pressure [GPa]

pySALEPlot: tracer.py

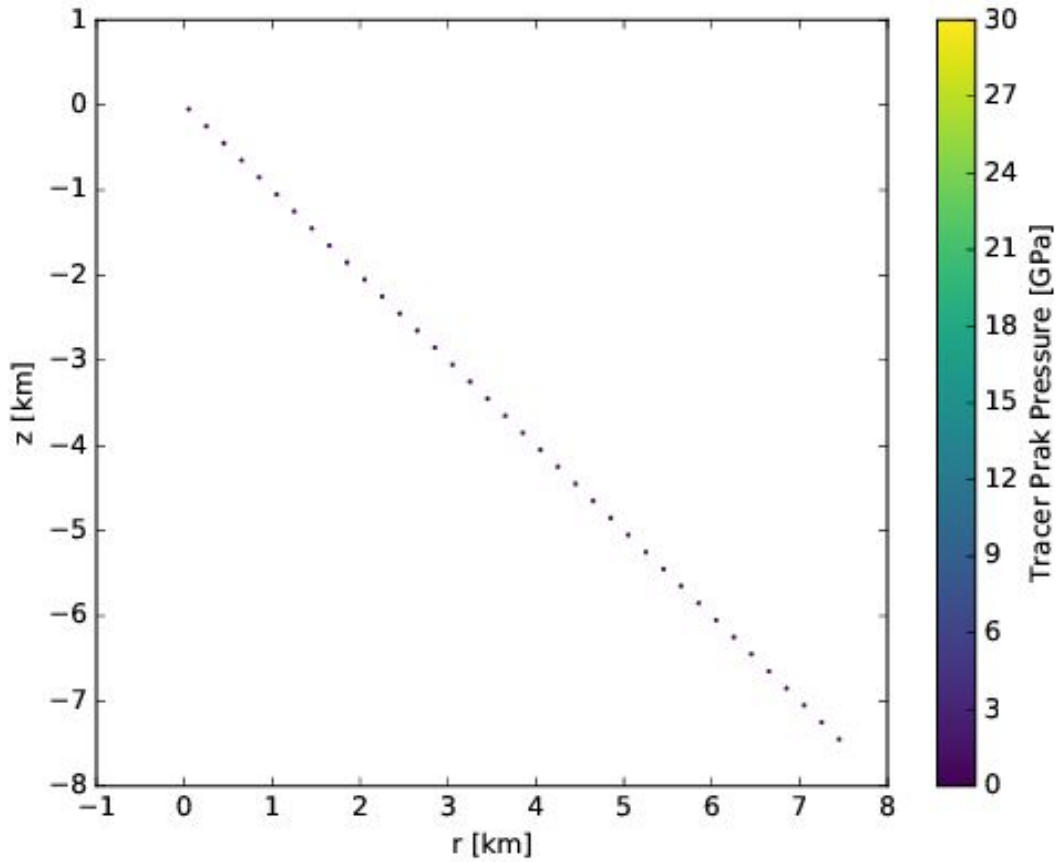
2. ある角度での最大圧力をプロットする
   a. トレーサー粒子の位置を得る
      i. XY座標: step.xmark と step.ymark
         1. オブジェクトの数: model.tracer_numu
         2. トレーサー粒子の番号:
            オブジェクトナンバーが0なら
            model.tru[0].start から model.tru[0].end まで

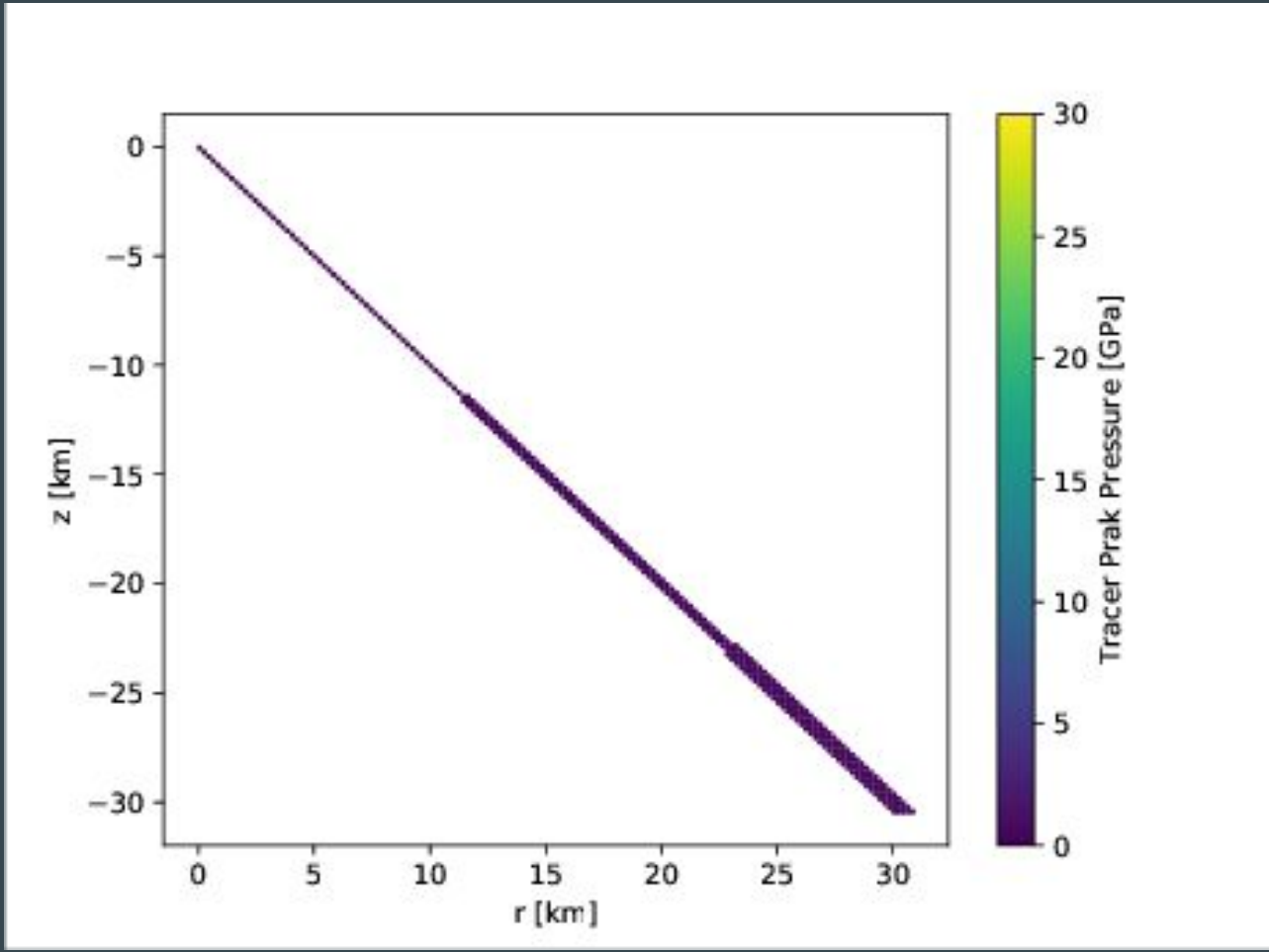
pySALEPlot: tracer.py

2. ある角度での最大圧力をプロットする
   b. トレーサー粒子の位置から角度を求める
      i. angles = np.rad2deg(np.arctan(step.xmark/step.ymark))
         -90 度 から 90度まで
   c. ある角度にある粒子を探す
      i. angle = -45.
      ii. angle_range = np.where((angles > angle - 0.5)
                & (angles < angle + 0.5))

pySALEPlot: tracer.py

2. ある角度での最大圧力をプロットする
   d. ある角度にある粒子だけをプロットする
      i. scat =
         ax.scatter(step.xmark[angle_range],
         step.ymark[angle_range],
         c=step.TrP[angle_range]*1e-9,
         vmin=0,vmax=30,cmap='viridis',s=4,linewidths=0)


z [km]
r [km]
Tracer Prak Pressure [GPa]
1
0
−1
−2
−3
−4
−5
−6
−7
−8
−1
0
1
2
3
4
5
6
7
8
30
27
24
21
18
15
12
9
6
3
0


0
−5
−10
−15
−20
−25
−30
z [km]
0
5
10
15
20
25
30
r [km]
30
25
20
15
10
5
0
Tracer Prak Pressure [GPa]

pySALEPlot: R_TrP.py

- R_TrP.py

3. ある角度での
   距離と最大圧力の関係を描く
   a. 角度を求める
   b. ある角度を探す
   c. ある角度だけプロットする
      i. ある時間での最大圧力をプロットする
         （自習課題）

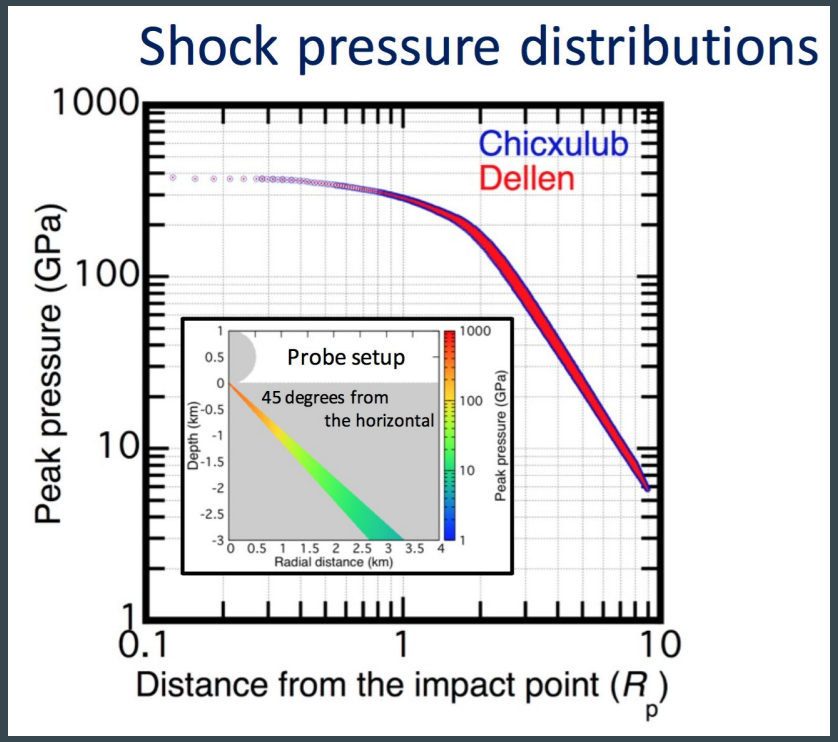


図:黒澤校長より寄贈

pySALEPlot: R_TrP.py

3. ある角度での
   距離と最大圧力の関係を描く
   d. 距離を求めて、並べる

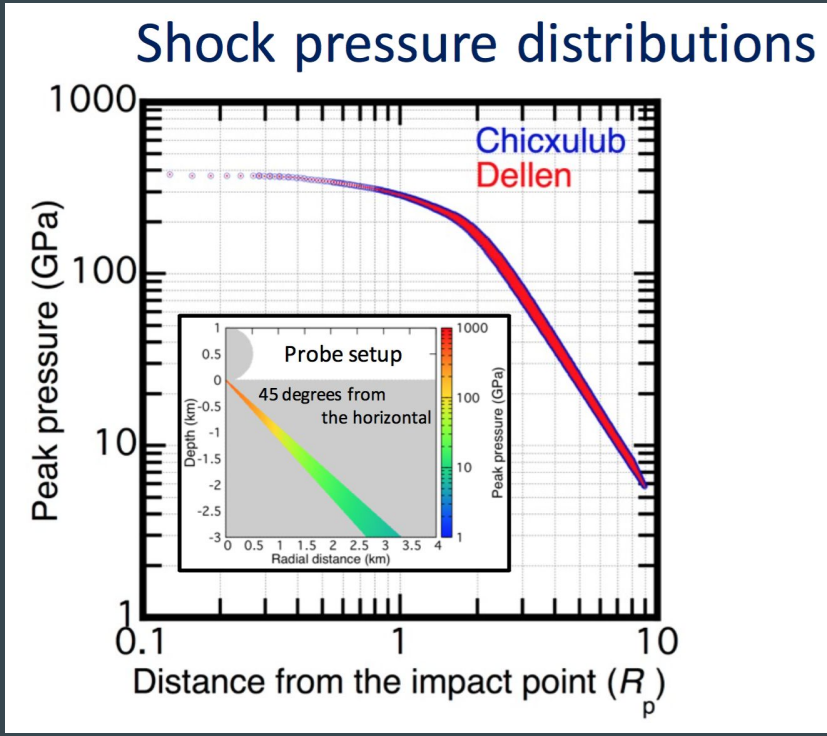
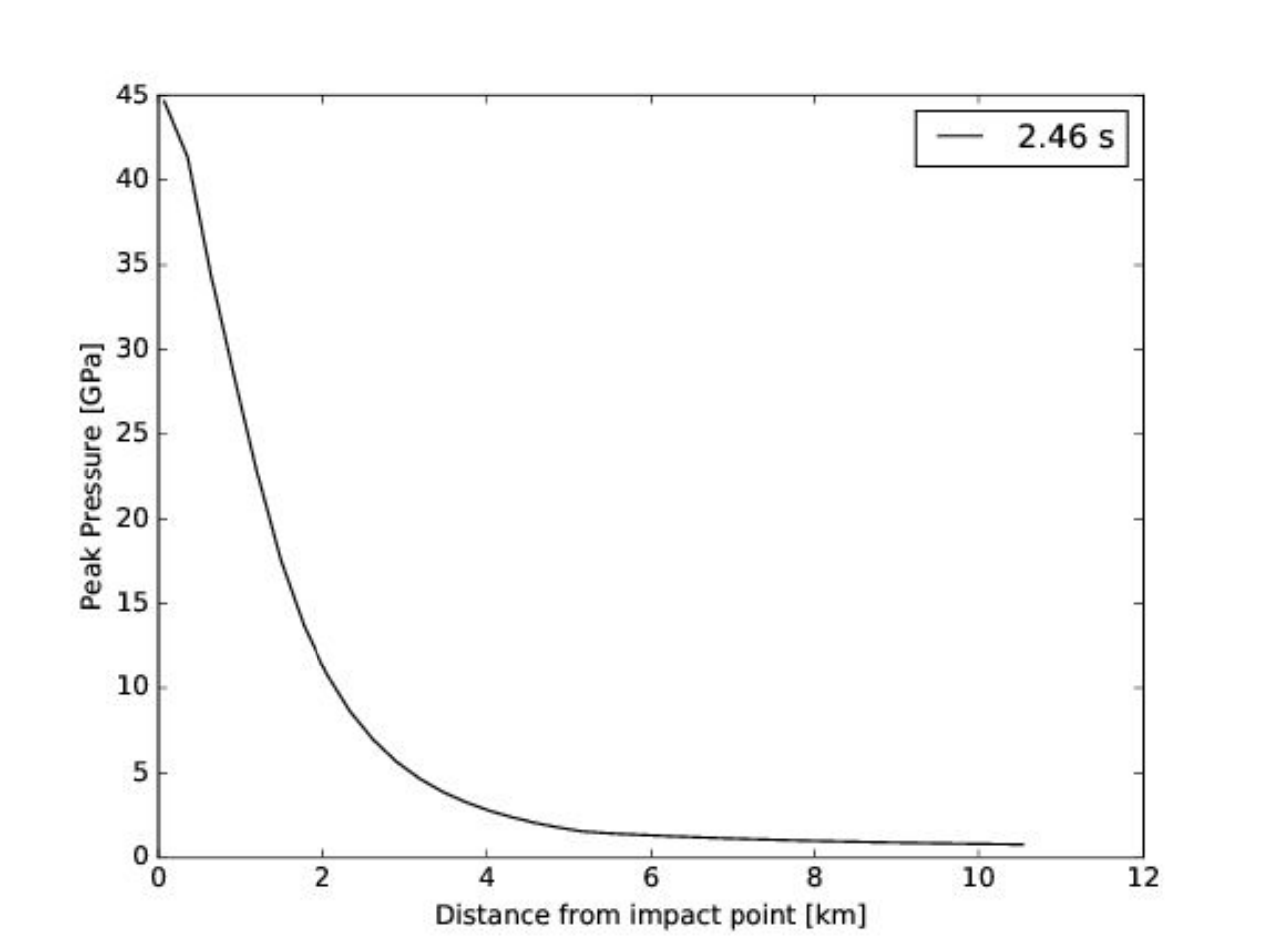


図:黒澤校長より寄贈

```
r = np.sqrt(step0.xmark[angle_range]**2
      +step0.ymark[angle_range]**2)
trp = step1.TrP[angle_range]*1e-9
ax.plot(np.sort(r),trp[np.argsort(r)])
```


2.46 s
Peak Pressure [GPa]
Distance from impact point [km]
0
5
10
15
20
25
30
35
40
45
0
2
4
6
8
10
12

まとめ

- pySALEPlotを用いた描画
  - 実践編
    - Sampleのpythonスクリプトを変更する
      - 出力したい変数に変える
        - Pre, Den, Tmp, ...
      - トレーサー粒子を描き出す
        - TrP, xmark, ymark, ...
      - 論文のために numpy なども使ってみる
        - np.sqrt, np.rad2deg, ...