中級課題編

…

中級課題

1. 衝撃加熱度評価
    - 横軸: 最大経験温度
      縦軸: 累積トレーサー粒子数(or 累積質量)
2. 衝撃圧力分布
    - 横軸: 衝突点からの距離
      縦軸: 最大経験圧力
        - 実践編を参照

中級課題

3. 掘削の流跡線描画
    - 計算空間内でのトレーサー粒子の軌跡
4. 衝撃波伝播の様子の可視化
    - 衝撃波の伝播の等時刻線の表示

中級課題進め方

1. 必要ファイルをDLする.
   Files_Intermediate.zip

2. ./examples/demo2Dをコピーし, 適当な名前をつける.
   例. cp –a demo2D Intermediate2D

3. Intermediate2Dに必要ファイルをアップロードし,
   解凍する. asteroid.inpとmaterial.inpは上書きして構わない.

4. iSALE2Dを実行する.

入力ファイル –material.inp-

```
#ISMAT ! iSale material input file identification string
-----------------------------------------------------------------------------
MATNAME     Material name                                   : proj___ : target_
EOSNAME     EOS name                                        : granit2 : granit2
EOSTYPE     EOS type                                        : aneos   : aneos
STRMOD      Strength model                                  : HYDRO   : HYDRO
DAMMOD      Damage model                                    : NONE    : NONE
ACFL        Acoustic fluidisation                           : NONE    : NONE
PORMOD      Porosity model                                  : NONE    : NONE
THSOFT      Thermal softening                               : NONE    : NONE
LDWEAK      Low density weakening                           : NONE    : NONE
-----------------------------------------------------------------------------
POIS        pois                                            : 0.5 : 0.5
<<END ! used to identify the end of this file
```

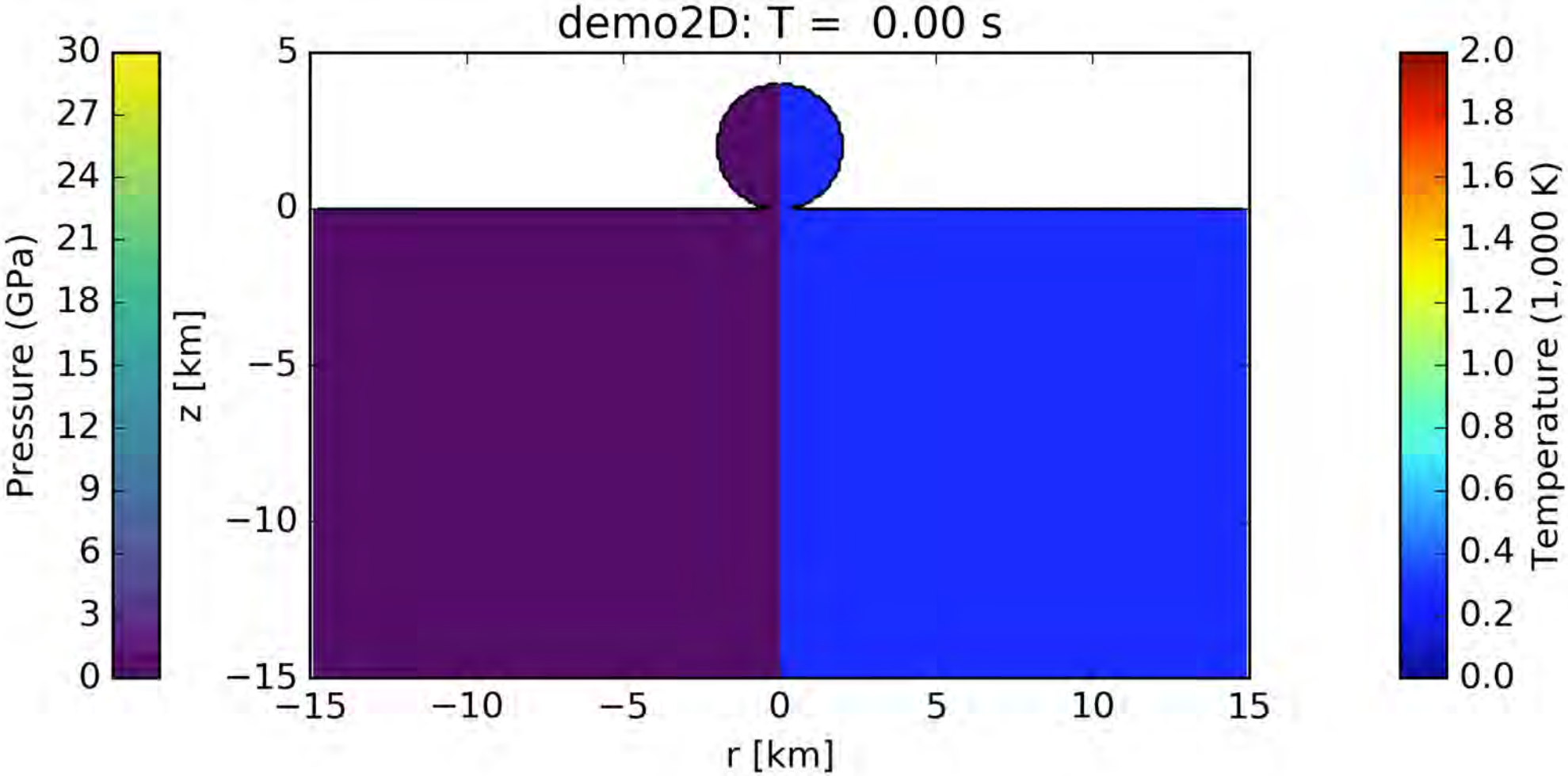
- ImpactorとTargetは同じ物質(granit2)だが, 計算中で区別される.
- 簡単のため強度なしの流体計算に変更.

入力ファイル –asteroid.inp-

```
------------------- General Model Info ---------------------------------
VERSION                 __DO NOT MODIFY__          : 4.1
DIMENSION               dimension of input file    : 2
PATH                    Data file path             : ./
MODEL                   Modelname                  : Intermediate2D
------------------- Mesh Geometry Parameters ---------------------------
GRIDH                   horizontal cells           : 0         : 100       : 30
GRIDV                   vertical cells             : 30        : 200       : 20
GRIDEXT                 ext. factor                : 1.03d0
GRIDSPC                 grid spacing               : 100.D0
GRIDSPCM                max. grid spacing          : -20.D0
------------------- Global Setup Parameters ----------------------------
S_TYPE                  setup type                 : DEFAULT
ALE_MODE                ALE modus                  : EULER
T_SURF                  Surface temp               : 293.D0
GRAV_V                  gravity                    : -9.81D0
GRAD_TYPE               gradient type              : DEFAULT
GRAD_DIM                gradient dimension         : 2
------------------- Projectile Parameters ------------------------------
OBJNUM                  number of proj.            : 1
OBJRESH                 CPPR horizontal            : 20
OBJVEL                  object velocity            : -12.0D3
OBJMAT                  object material            : proj___
OBJTYPE                 object type                : SPHEROID
OBJTPROF                object temp prof           : CONST
```

```
------------------- Target Parameters ----------------------------------
LAYNUM                  number of layers           : 1
LAYPOS                  layer position             : 150
LAYMAT                  layer material             : target_
LAYTPROF                layer therm. prof          : CONST
------------------- Time Parameters ------------------------------------
DT                      initial time increment     : -0.01
DTMAX                   maximum timestep           : -0.1
TEND                    end time                   : -10.D0
DTSAVE                  save interval              : -0.1D0
------------------- Boundary Conditions --------------------------------
BND_L                   left                       : FREESLIP
BND_R                   right                      : OUTFLOW
BND_B                   bottom                     : NOSLIP
BND_T                   top                        : OUTFLOW
------------------- Numerical Stability Parameters ---------------------
AVIS                    art. visc. linear          : 0.24D0
AVIS2                   art. visc. quad.           : 1.2D0
------------------- Tracer Particle Parameters -------------------------
TR_QUAL     integration qual.          : 1
TR_SPCH     tracer spacing X           : -1.D0 : -1.D0
TR_SPCV     tracer spacing Y           : -1.D0 : -1.D0
TR_VAR      add. tracer fiels          : #TrP-TrT-Trd-Trp-Trt#
------------------- Data Saving Parameters -----------------------------
QUALITY                 Compression rate           : -50
VARLIST                 List of variables          : #Den-Pre-Tmp-Yld-Dam-Ert-Vib-YAc-PVb-VEL#
------------------------------------------------------------------------
```

- 計算領域を微修正.
- OBJMATとLAYMATをmaterial.inpと対応.
- トレーサ粒子を挿入.


demo2D: T = 0.00 s
Pressure (GPa)
30
27
24
21
18
15
12
9
6
3
0
z [km]
5
0
−5
−10
−15
−15
−10
−5
0
5
10
15
r [km]
Temperature (1,000 K)
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0

中級課題

- 共通する前提
  - 使用するファイルにtracer粒子が含まれていること
    - 物理量: TrT, TrP, Trd, Trp, Trt
      - ???という方は初級課題１小課題1–4に戻ること
  - もとになるscriptからスタートしてもOK.
    腕に自信のある方はイチから作成してみましょう.
- 書いてあるのは一例にすぎません
  - より良い方法があれば、そちらで構いません.
- 例図は、demo2Dにtracer粒子を加えたものです.

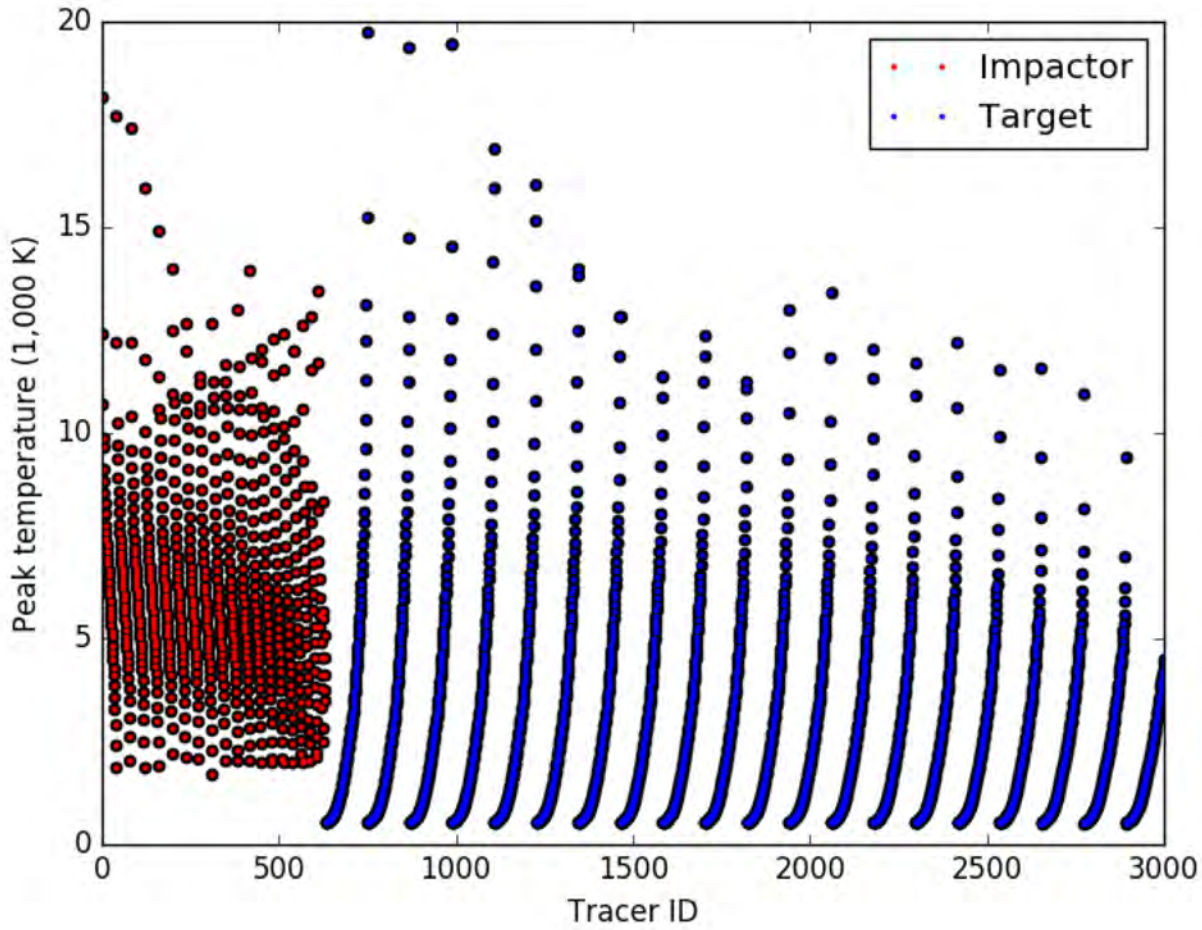
1. 衝撃加熱度評価

- 横軸: 最大経験温度
  縦軸: 累積トレーサー粒子数(or 累積質量)

  もとになるスクリプト: Number-TrT.py
  トレーサ番号に対して, 最大経験温度を描画

  解答例: ./isale2019cfca/TrT_Num.py

1. 衝撃加熱度評価 Number-TrT.py



1. 衝撃加熱度評価

Number-TrT.pyで行っている処理

1. 読み込むデータの種類、時間
    a. step=model.readStep('TrT', model.laststep)
2. plotに使用する他の値
    a. Num_impactor = model.tru[0].end
        Projectileにあるtracer粒子数で規格化する場合
3. plotに使う関数
    a. ax.plot(x, y, ’k’)

1. 衝撃加熱度評価

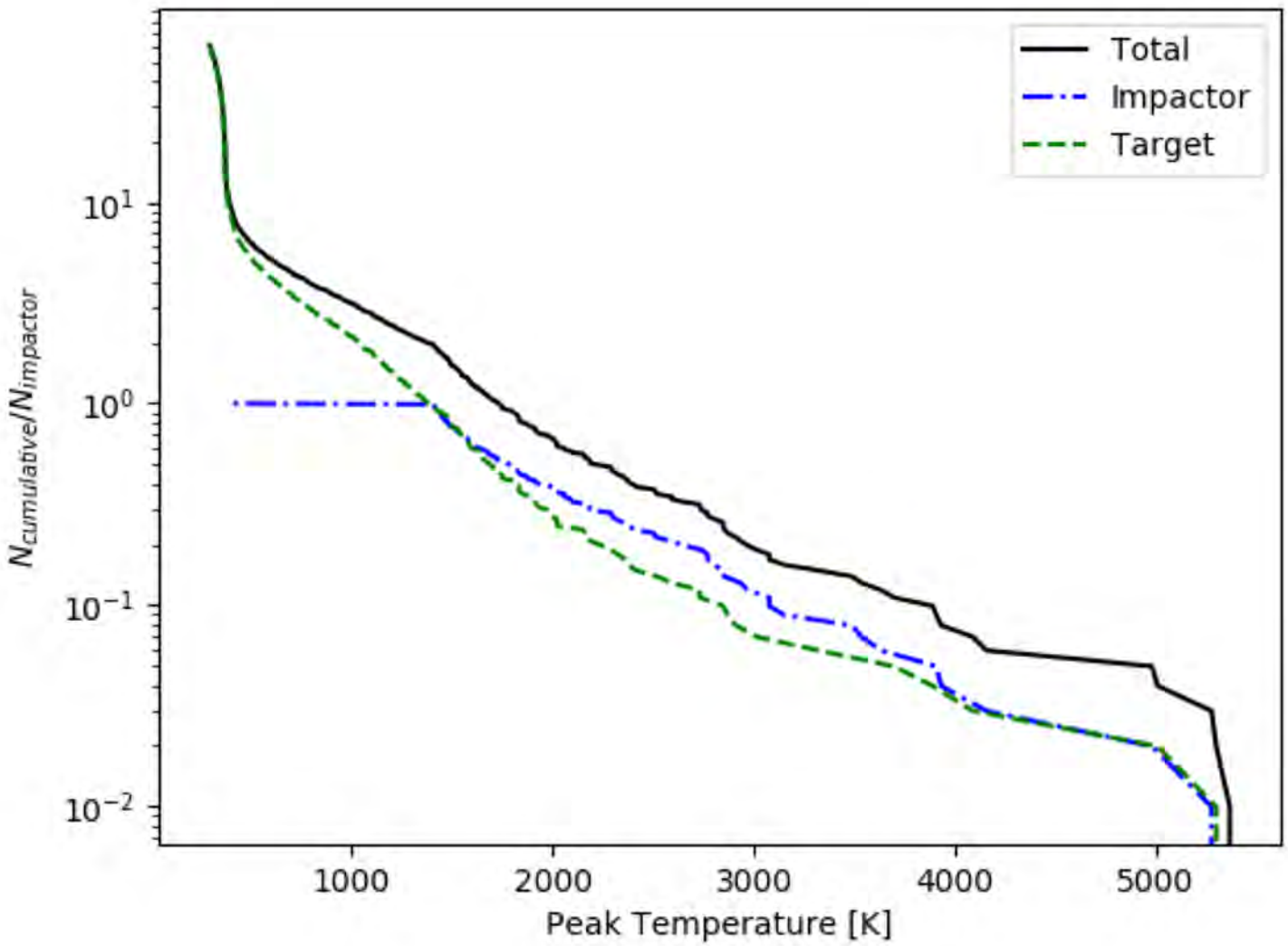
Plotに使う関数 “ax.plot(x, y, ’k’)”

x: step.TrTを高い温度から低い温度に並び替える.

hint: np.sort()

y: 対応するstep.TrTの数を規格化する数で割る.

hint: np.arange() + len(x)

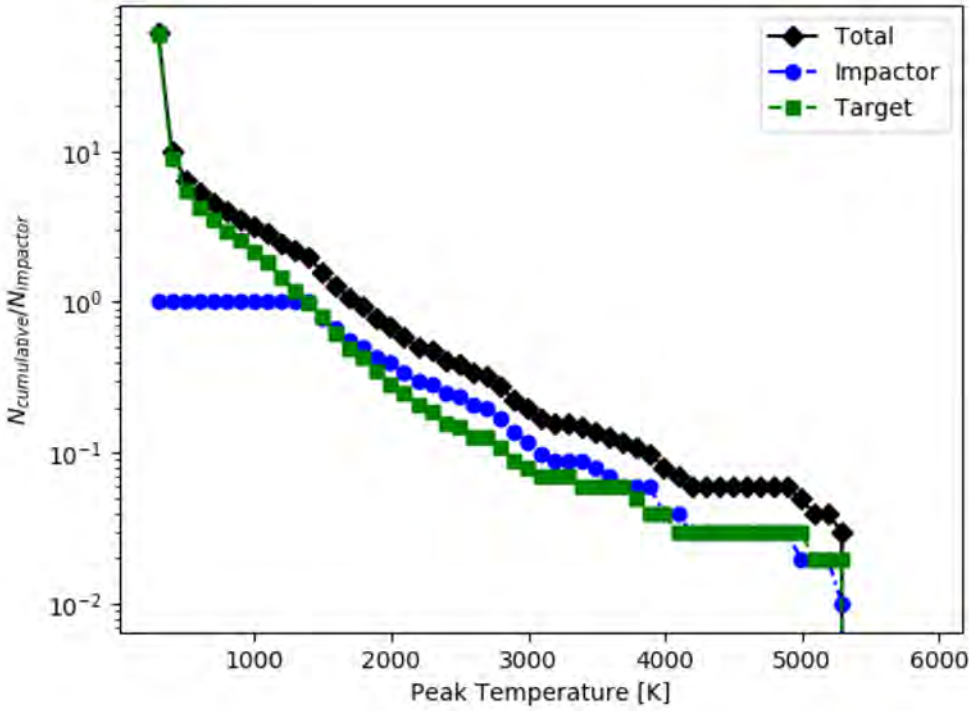

Total
Impactor
Target
10^1
10^0
10^{-1}
10^{-2}
$N_{cumulative}/N_{impactor}$
1000
2000
3000
4000
5000
Peak Temperature [K]

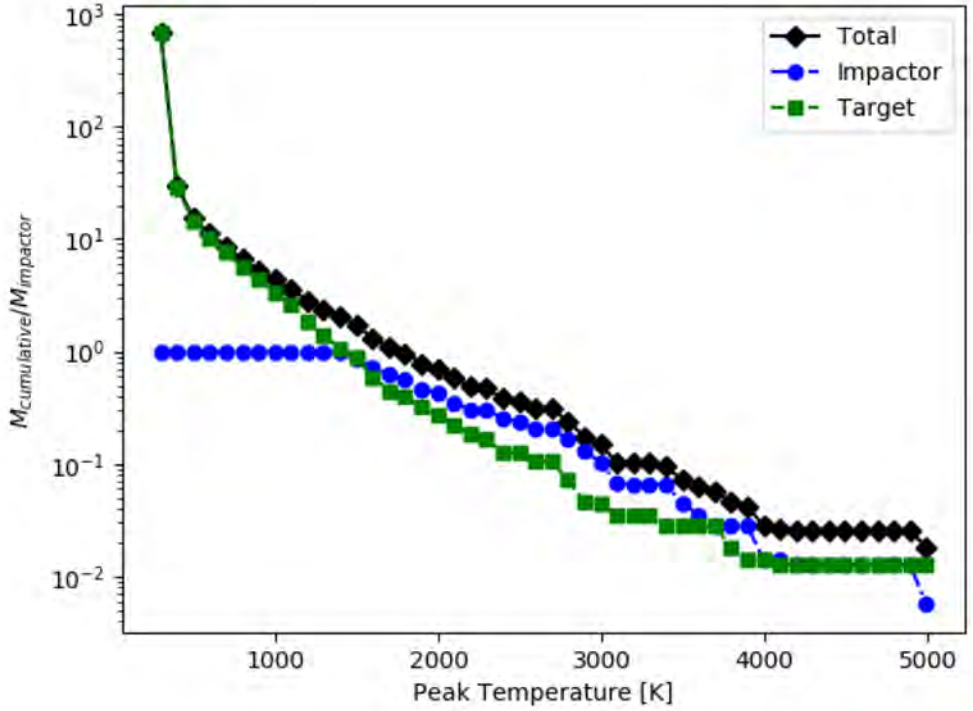
1. 衝撃加熱度評価

上級編

1. Impactorとtarget毎に分ける
   hint: step.TrTをどう読み込む？
2. ”bin”毎にstep.TrTを並び替える
   hint: pd.Series()+pd.cut()
3. 質量で規格化する
   対応するstep.TrT毎の質量を求める
   hint: examples/Collision2D/Plotting/ShockP.py

1. 衝撃加熱度評価


Total
Impactor
Target
$N_{cumulative}/N_{impactor}$
10^{1}
10^{0}
10^{-1}
10^{-2}
1000
2000
3000
4000
5000
6000
Peak Temperature [K]


Total
Impactor
Target
$M_{cumulative}/M_{impactor}$
10^{3}
10^{2}
10^{1}
10^{0}
10^{-1}
10^{-2}
1000
2000
3000
4000
5000
Peak Temperature [K]

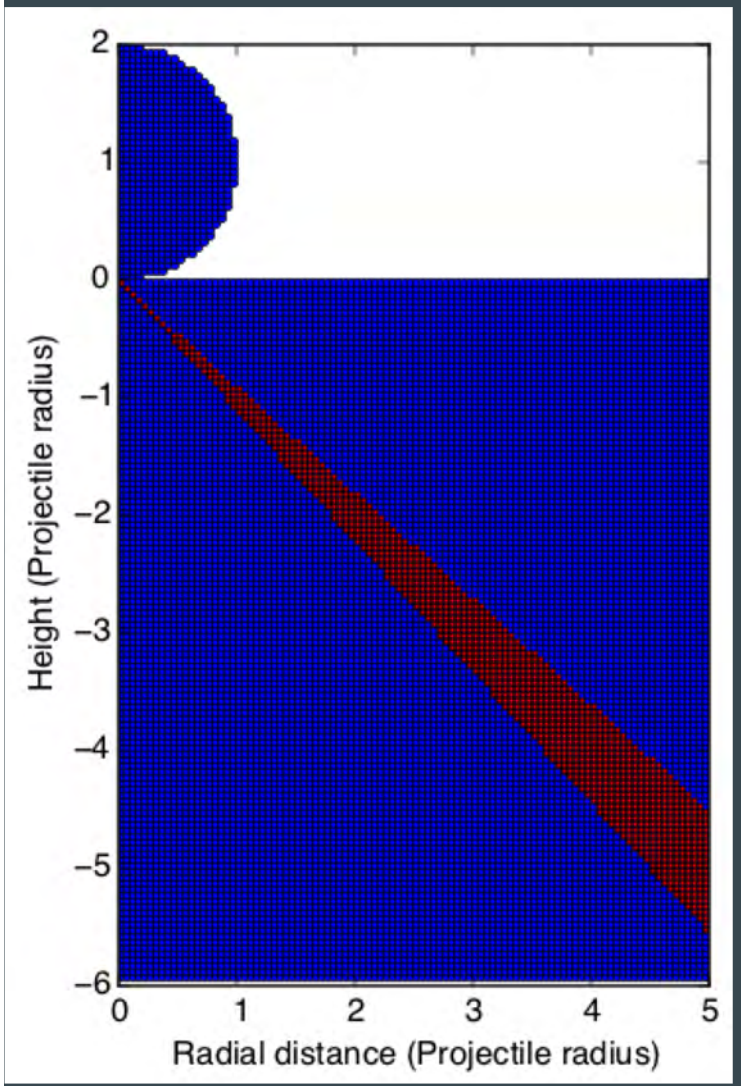
2. 衝撃圧力分布

- 横軸: 衝突点からの距離
  縦軸: 最大衝撃圧力

  もとになるスクリプト: R-Z_w_tracer.py
  トレーサ探針の描画

  解答例: ./R_TrP.py

2. 衝撃圧力分布



衝突点から斜め45±3度に見下ろした
トレーサを取り出すことができた.

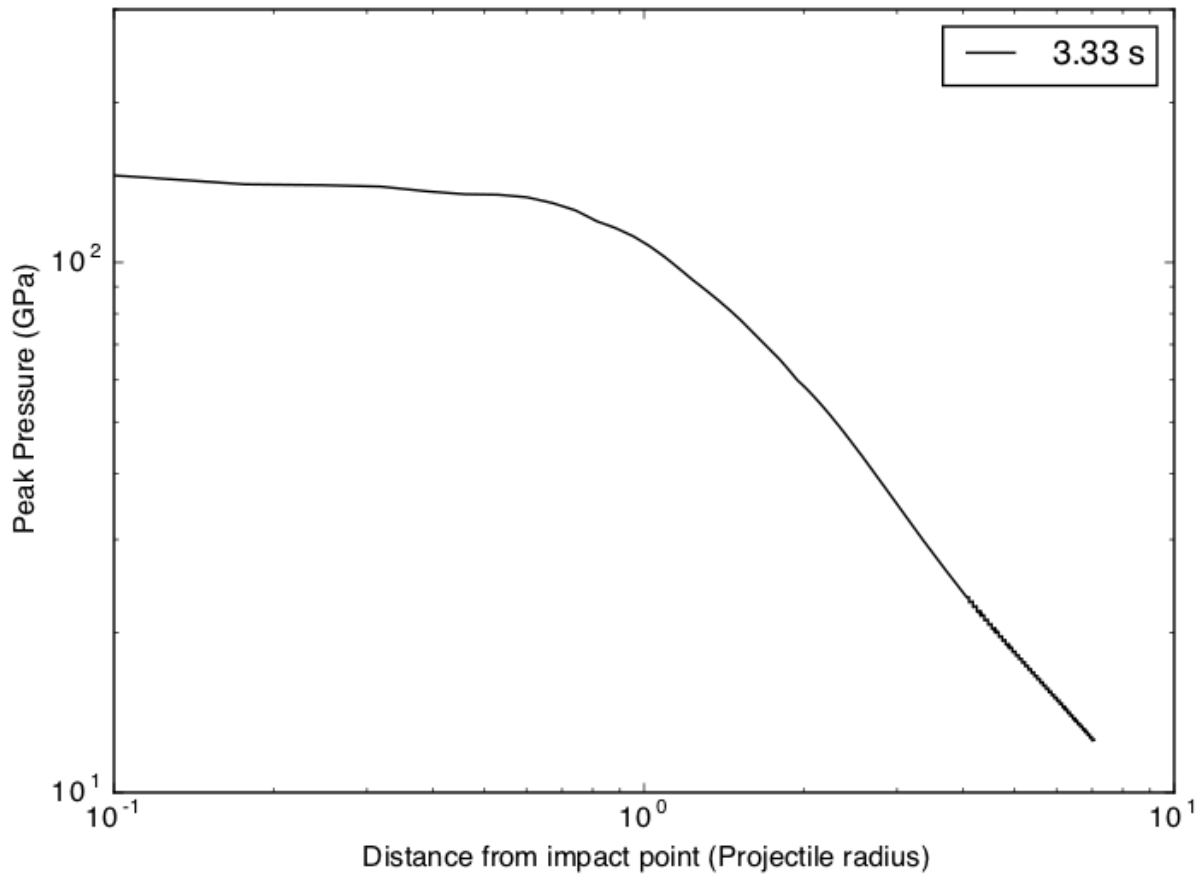
後は...

1. 各トレーサの衝突点からの距離を計算.

2. 各トレーサの最大衝撃圧力を取得.

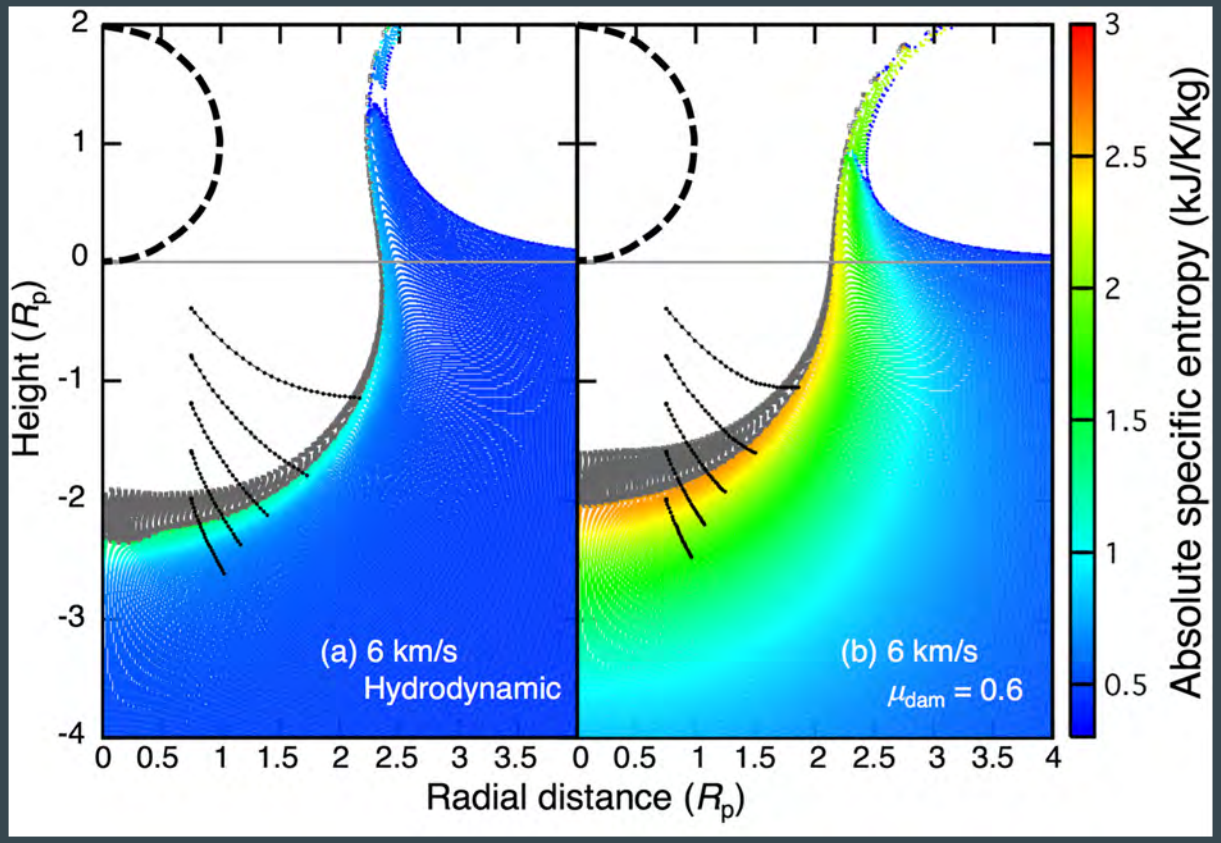
3. 距離に対して最大衝撃圧力をプロット

2. 衝撃圧力分布


3.33 s
Peak Pressure (GPa)
10^2
10^1
10^{-1}
10^0
10^1
Distance from impact point (Projectile radius)

3. 掘削の流跡線描画

- Figure 2 from Kurosawa and Genda 2018



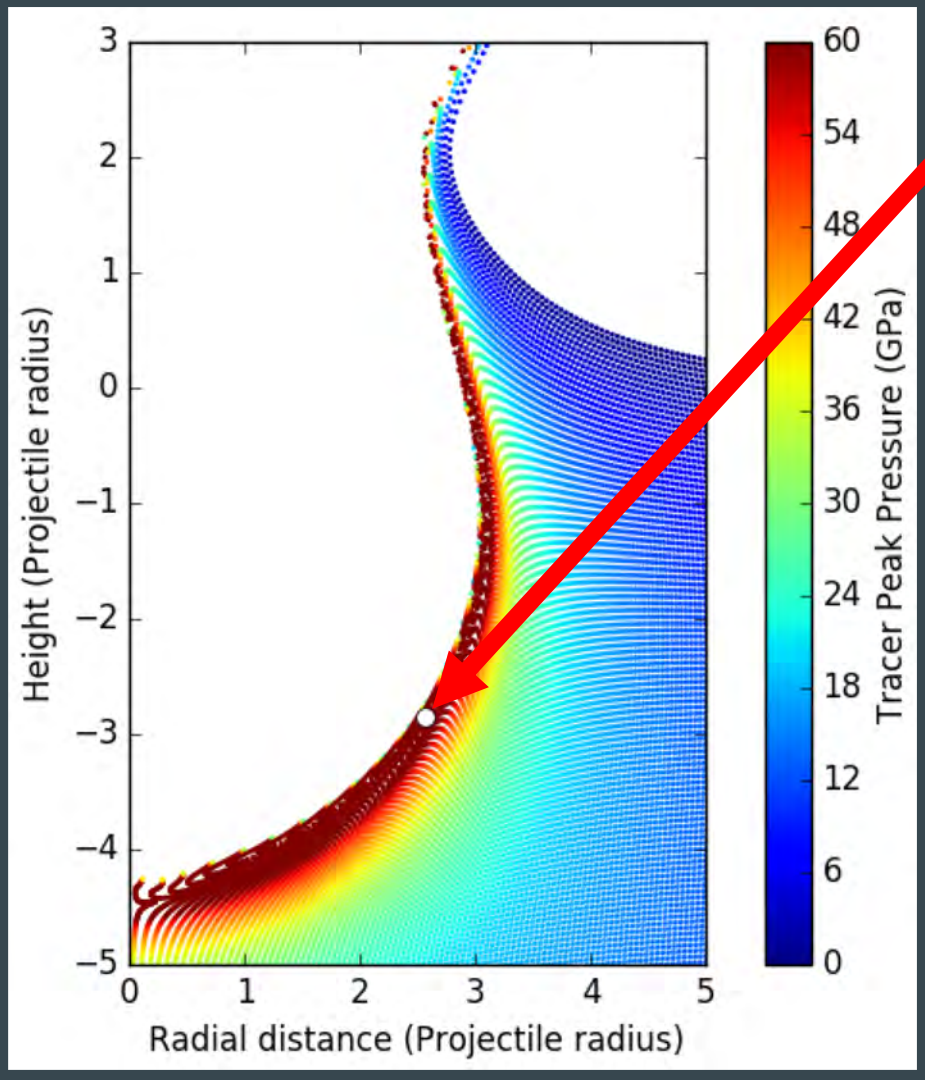
3. トレーサー粒子を使った流跡線の表示

- 計算空間内(*R*-*Z*)でのトレーサー粒子の軌跡

もとになるスクリプト: Selected_tracer.py
計算の最終ステップでの特定のトレーサ位置を描画.

解答例: ./tracer_line.py

3. トレーサー粒子を使った流跡線の表示



初期に(1 km, -1 km)に位置していたトレーサ

3. トレーサー粒子を使った流跡線の表示

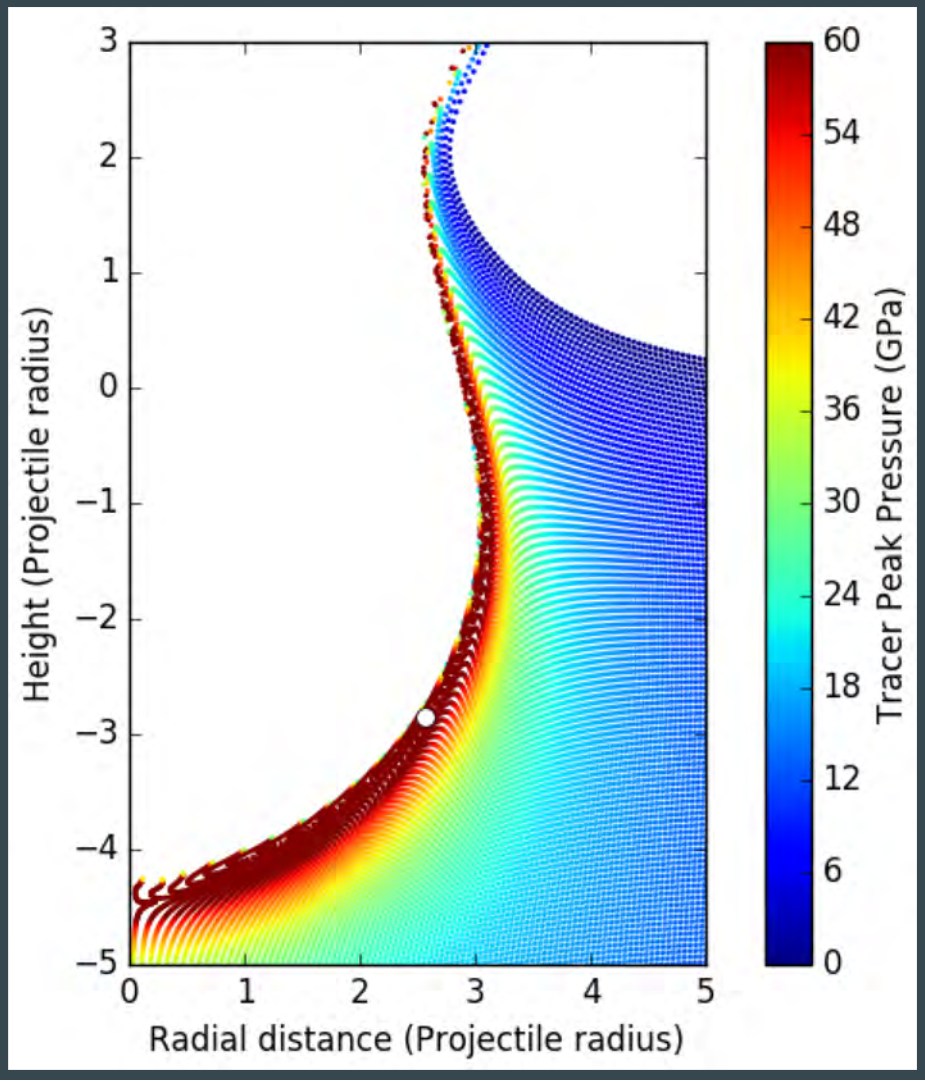
Selected_tracer.pyで行っている処理.

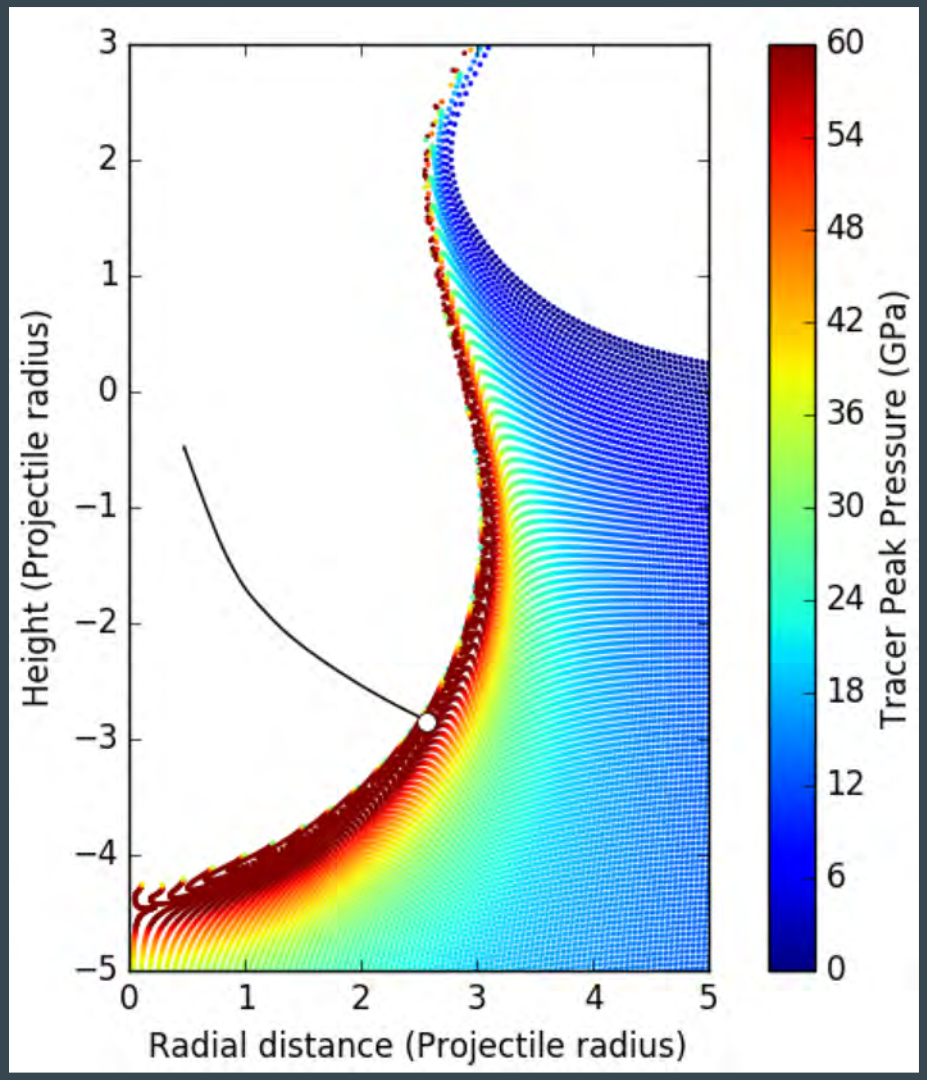
1. トレーサ粒子の最大衝撃圧力を読み込む
   (本課題については読み込むトレーサ物理量は何でもよい.)
   a. step=model.readStep('TrP', 0)
2. plotに使用するtracer 粒子のidを設定
   a. tracer_id = 250 (手動設定)
   b. tracer_id = step.findTracer(1.0,-1.0)
      (自動で(1,-1)付近のtracer粒子を探し出す)
      どちらか好きな方法で

3. トレーサー粒子を使った流跡線の表示

1. tracer粒子の軌跡を記録
   a. 記録する場所の設定
      tracer_x = [] など
   b. for文の中で値を追加
      tracer_x.append(???)
      hint: ???にはtracer粒子のidに応じたx座標の値がほしい
2. 記録した値をplot
   a. ax.plot(tracer_x, tracer_y, ‘k-’)

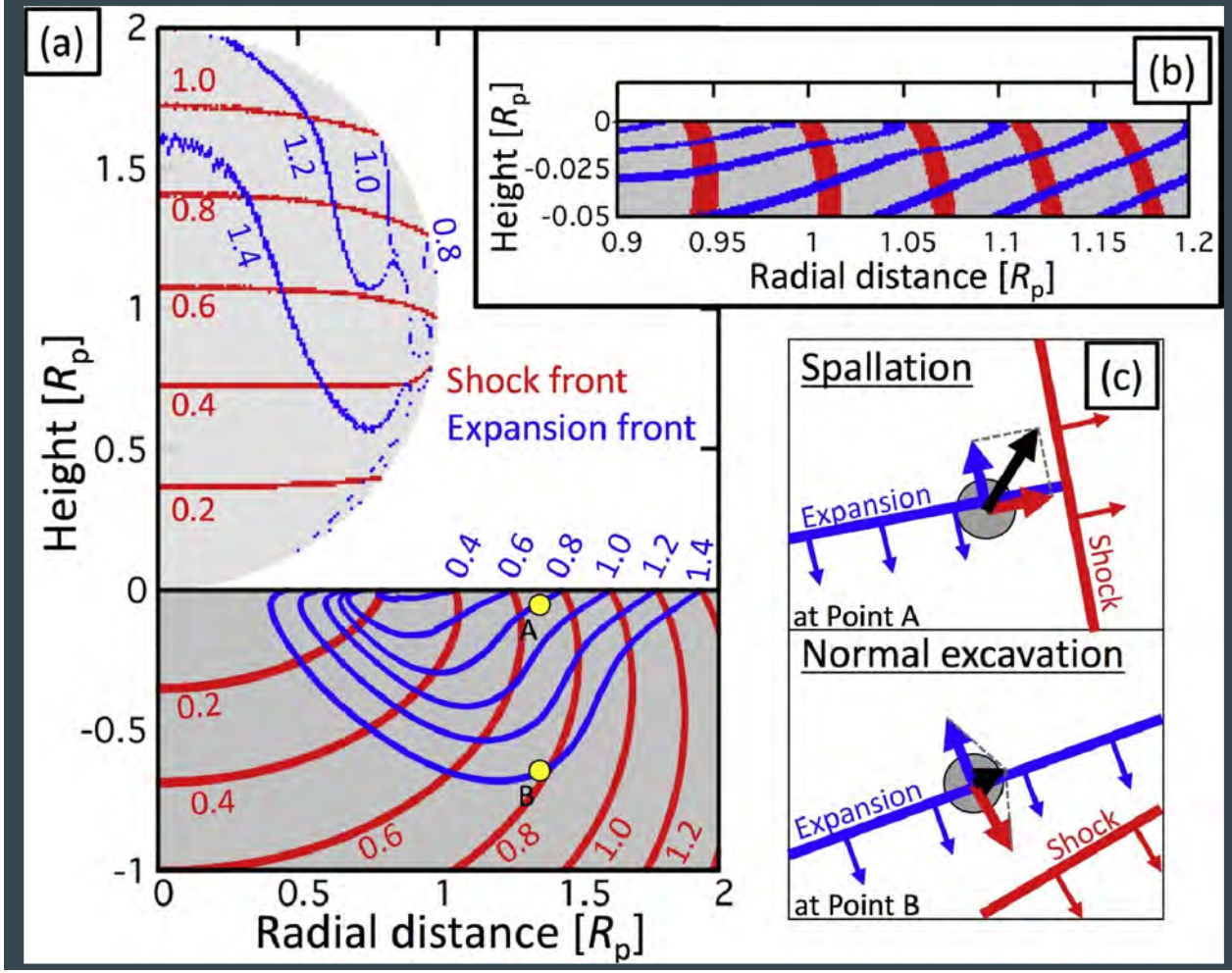
3. トレーサー粒子を使った流跡線の表示





4. 衝撃波伝播の様子の可視化

- Figure 8 from Kurosawa et al. 2018



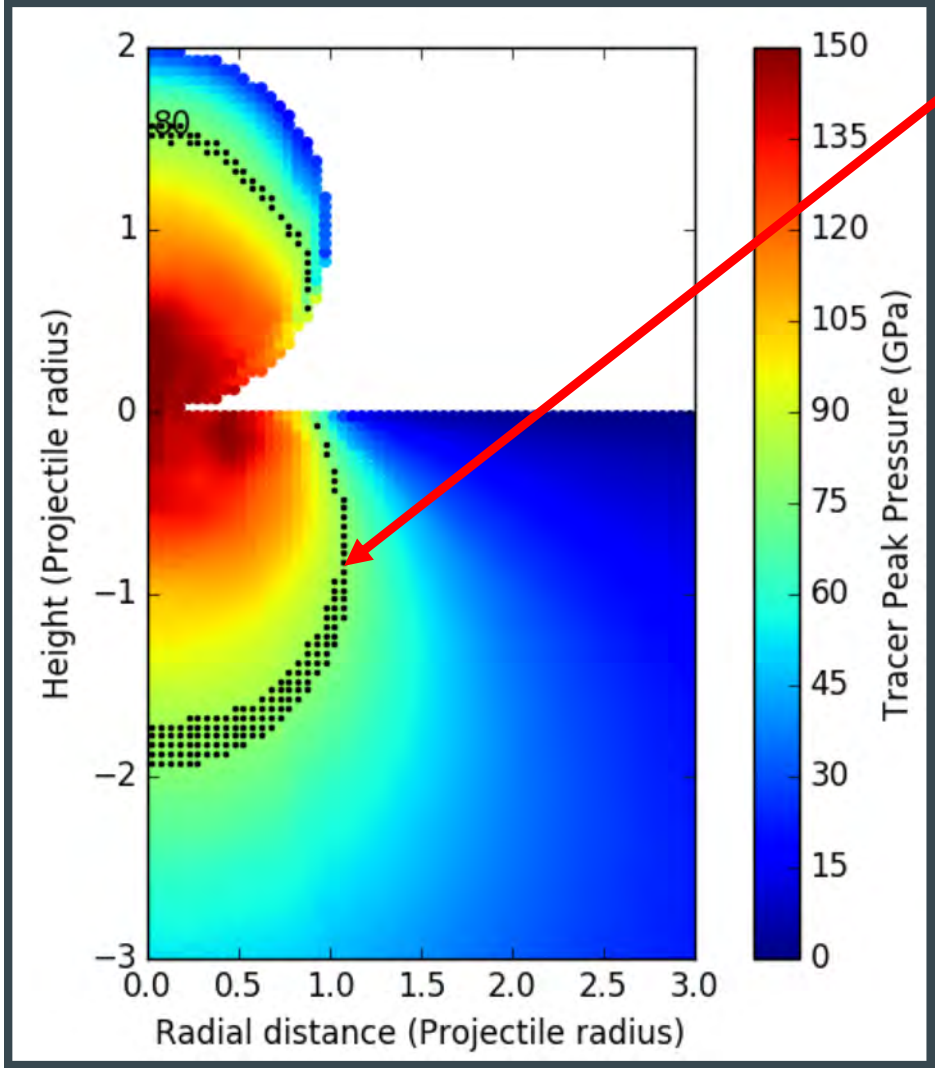
4. 衝撃波伝播の様子の可視化

- 衝撃波の伝播の等時刻線の表示

もとになるスクリプト: PeakP_provenance.py
トレーサ粒子の最大経験圧力を初期位置に対して描画.

解答例: ./isale2019cfca/isochrone.py

4. 衝撃波伝播の様子の可視化



80 GPaの等圧線

4. 衝撃波伝播の様子の可視化

PeakP_provenance.pyで行っている処理.

1. 読み込むデータの種類、時間.
   初期位置の情報と計算の最終ステップの最大衝撃圧力が必要.

```
step0=model.readStep('TrP', 0)
step  =model.readStep('TrP', model.laststep)
```

2. 最大衝撃圧力のカラーバープロット ax.scatter

3. 等圧線の表示 np.where

4. 衝撃波伝播の様子の可視化

1. 読み込むデータの種類、時間
   a. step=model.readStep('Trp', i),
   forループで全計算ステップのTrpを読み込む.
2. tracer粒子の圧力がしきい値を超えた時間を記録する場所を用意してから、記録する
   a. shock_time = np.zeros(model.tracer_num)
   b. if(step.TrP[tracer] >= shock_pressure):
      shock_time[tracer] = step.time
   hint: tracer粒子で判定するには何が必要？

4. 衝撃波伝播の様子の可視化

1. 記録された時間をplot
   a. ax.scatter(step0.xmark, step0.ymark, c=shock_time)

2. 等時刻線を描きたい場合
   a. shock_timeがある時刻間の粒子を探す
      hint: tracer粒子で角度を判定する際に使ったのは？

4. 衝撃波伝播の様子の可視化