iSALEトレーサ粒子解析サンプルプログラム

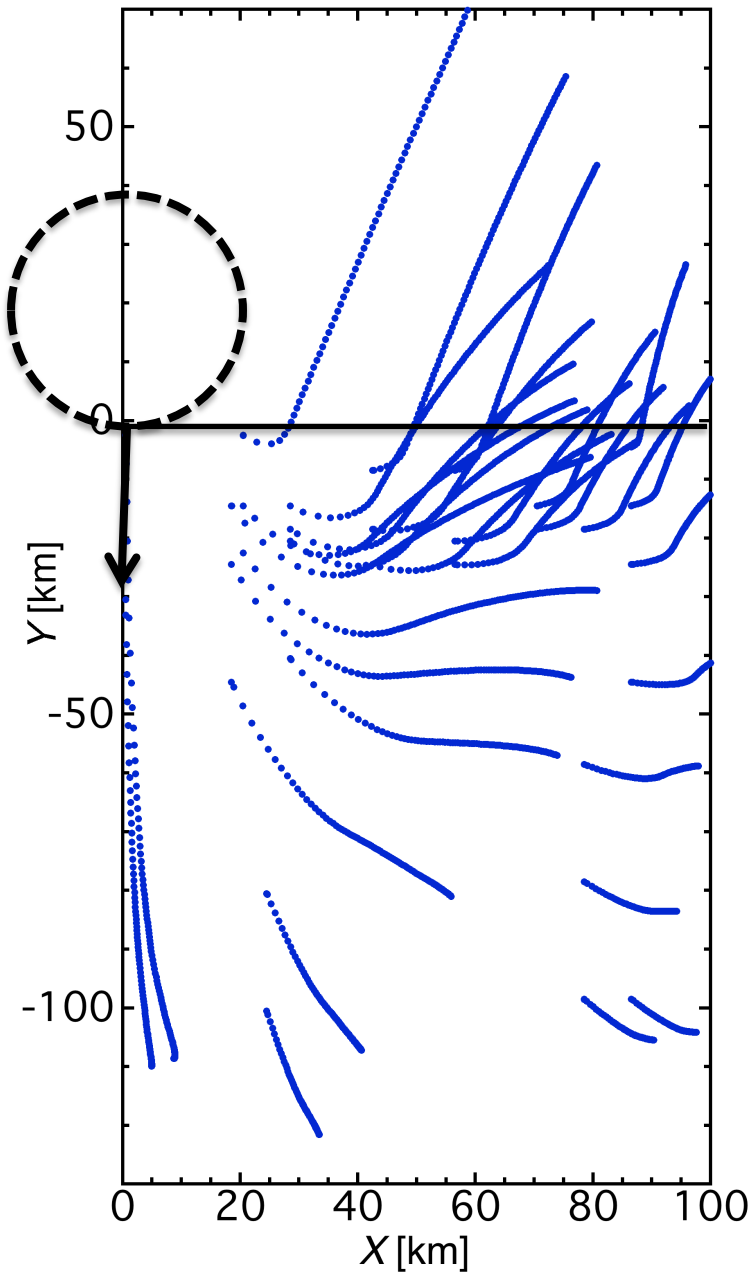
黒澤耕介, 千秋博紀
千葉工業大学 惑星探査研究センター

Special thanks: 脇田茂

2015 8/20

トレーサ粒子解析で何ができるか？

Lagrangian tracer particleとは

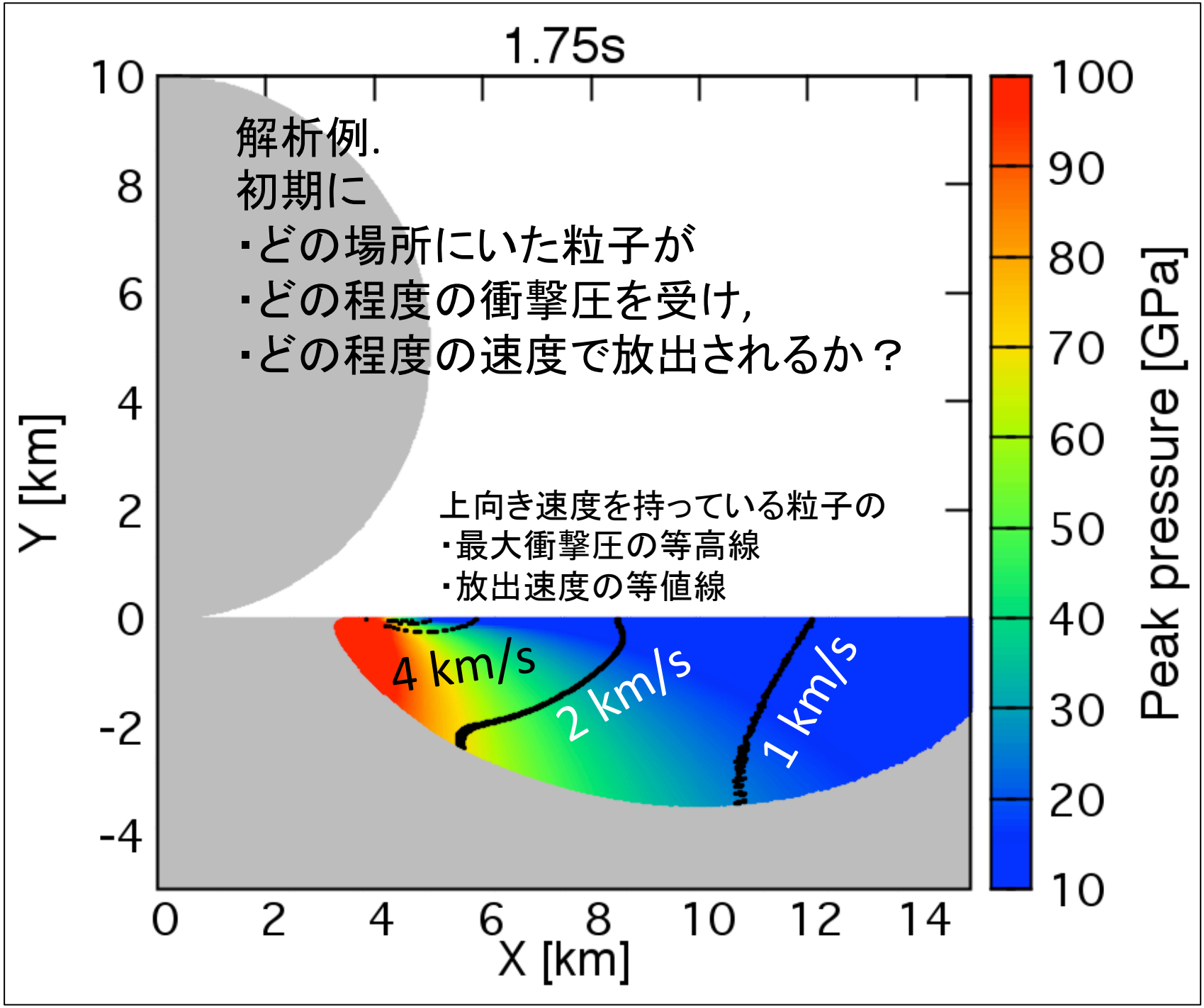


質量なしで流れと一体化する粒子

-> 格子法の計算でもある場所の
物質がどこまで動いたかを
追跡できる!

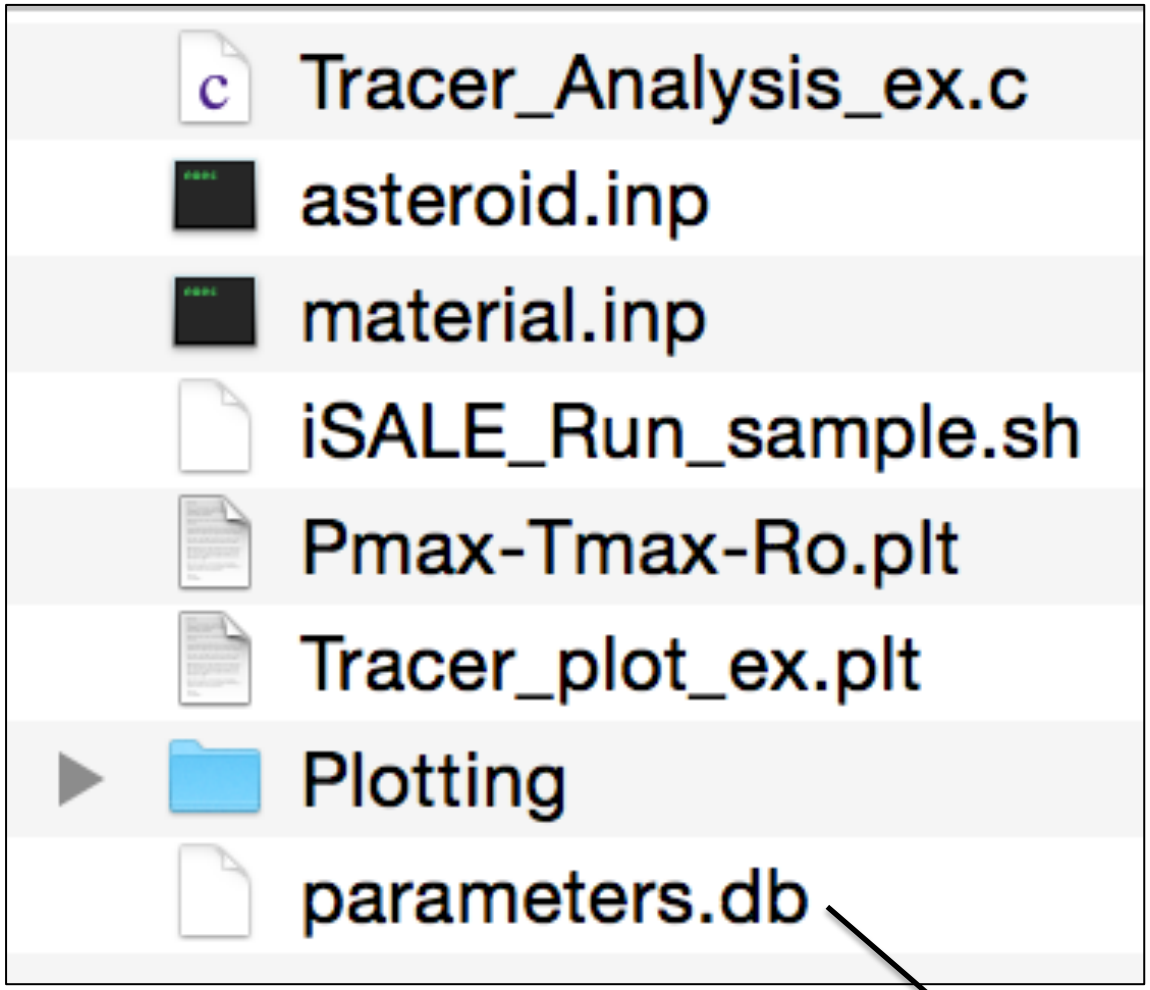
ボーナス

ある時刻, ある格子点における
温度圧力を記録できる!


1.75s
解析例.
初期に
・どの場所にいた粒子が
・どの程度の衝撃圧を受け,
・どの程度の速度で放出されるか？
上向き速度を持っている粒子の
・最大衝撃圧の等高線
・放出速度の等値線
4 km/s
2 km/s
1 km/s
Y [km]
X [km]
Peak pressure [GPa]
10
8
6
4
2
0
-2
-4
0
2
4
6
8
10
12
14
100
90
80
70
60
50
40
30
20
10

サンプルプログラムの使い方

Zipファイルの中身



<- トレーサ解析用 C言語プログラム

<- iSALEのインプットファイル

<- 計算-解析-描画を自動で
行わせるシェルスクリプト

<- 描画に用いるgnuplotの
スクリプト

<- トレーサ粒子を書き出す
iSALEPlotのインプットファイル

iSALEの各種パラメータの説明書

まずは使ってみましょう１

1. iSALEをインストールしたフォルダを下り
「examples」の中に適当な名前のフォルダを
新規作成. (以下「Sample2D」とします.)

2. Zipファイルを解凍し,「iSALE_sample_150820」内の8個の
ファイル＆フォルダを「Sample2D」にコピペ.

3. 「examples」内のサンプルプログラムの中から適当に
選んで(例えば「demo2D」)以下の６つのリンクを
「Sample2D」にコピペ

- eos
- iSALE2D
- iSALEMat
- iSALEPar
- iSALEPlot
- VIMoD

まずは使ってみましょう 2

ここまでで「Sample2D」内はこうなっているはず

```
kurosawakousuke-no-MacBook-Pro:Sample2D kosukekurosawa$ pwd
/Users/kosukekurosawa/iSALE-Chicxulub/install/share/examples/Sample2D
kurosawakousuke-no-MacBook-Pro:Sample2D kosukekurosawa$ ls
Plotting                asteroid.inp            iSALEPar                parameters.db
Pmax-Tmax-Ro.plt        eos                     iSALEPlot               vimod
Tracer_Analysis_ex.c    iSALE2D                 iSALE_Run_sample.sh
Tracer_plot_ex.plt      iSALEMat                material.inp
kurosawakousuke-no-MacBook-Pro:Sample2D kosukekurosawa$
```

4. ターミナルで「Sample2D」まで移動し以下のコマンドを打つ.

```
kurosawakousuke-no-MacBook-Pro:Sample2D kosukekurosawa$ chmod +x iSALE_Run_sample.sh
kurosawakousuke-no-MacBook-Pro:Sample2D kosukekurosawa$ ls -l iSALE_Run_sample.sh
-rwxr-xr-x@ 1 kosukekurosawa  staff  689  8 20 10:40 iSALE_Run_sample.sh
kurosawakousuke-no-MacBook-Pro:Sample2D kosukekurosawa$
```

-rwxr-xr-xとなっていればOK

※ chmod +x はスクリプトに実行権限を与えるコマンド.

環境によっては入力不要の場合もある.

まずは使ってみましょう 3

4. 以下のコマンドを打ち, スクリプトを実行.
   ./iSALE_Run_sample.sh

```
kurosawakousuke-no-MacBook-Pro:Sample2D kosukekurosawa$ ./iSALE_Run_sample.sh

  +++++++++++++++++++++++++++++++++++++++++++++++
  +++                  iSALE                  +++
  +++    by Kai Wuennemann, Gareth Collins    +++
  +++          and Dirk Elbeshausen           +++
  +++                                         +++
  +++  based on SALEB  by Ivanov              +++
  +++           SALES  by Melosh              +++
  +++           SALE   by Amsden et al.       +++
  +++                                         +++
  +++++++++++++++++++++++++++++++++++++++++++++++

Opening parameter input-file.....................................asteroid.inp
Opening material input-file......................................material.inp
Your input file is up to date!..........................No changes required!
Checking input-file version.................................................OK
Checking non-optional parameters; errors...................................none
Checking list of valid values; errors......................................none
```

iSALE2D, iSALEPlot, Cの解析プログラム, gnuplotが次々に実行され,
計算, 解析, 描画が10分ほどで完了する.

計算出力

・「Sample2D」内に生成された「Processed」内に解析結果が格納されている.

「./Processed/Data」
テキスト形式. カレイダグラフなどのソフトでもスペース区切りで開くことができる.

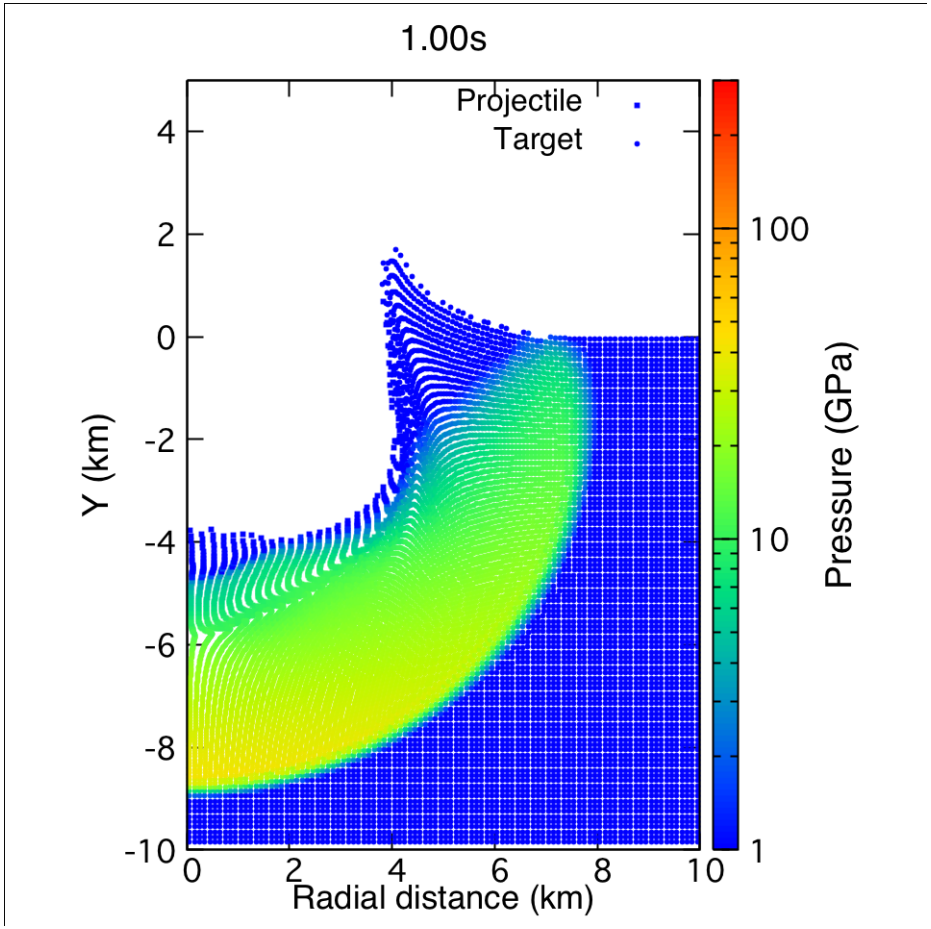
「./Processed/Figures_png」&「./Processed/Figures_eps」
gnuplotによる画像出力.
各時刻に一枚の図が出力されているので, gifアニメなど動画へも加工も可能

※iSALE-2Dによるバイナリデータ「jdata.dat」は従来通り「./Sample2D」に格納されている.

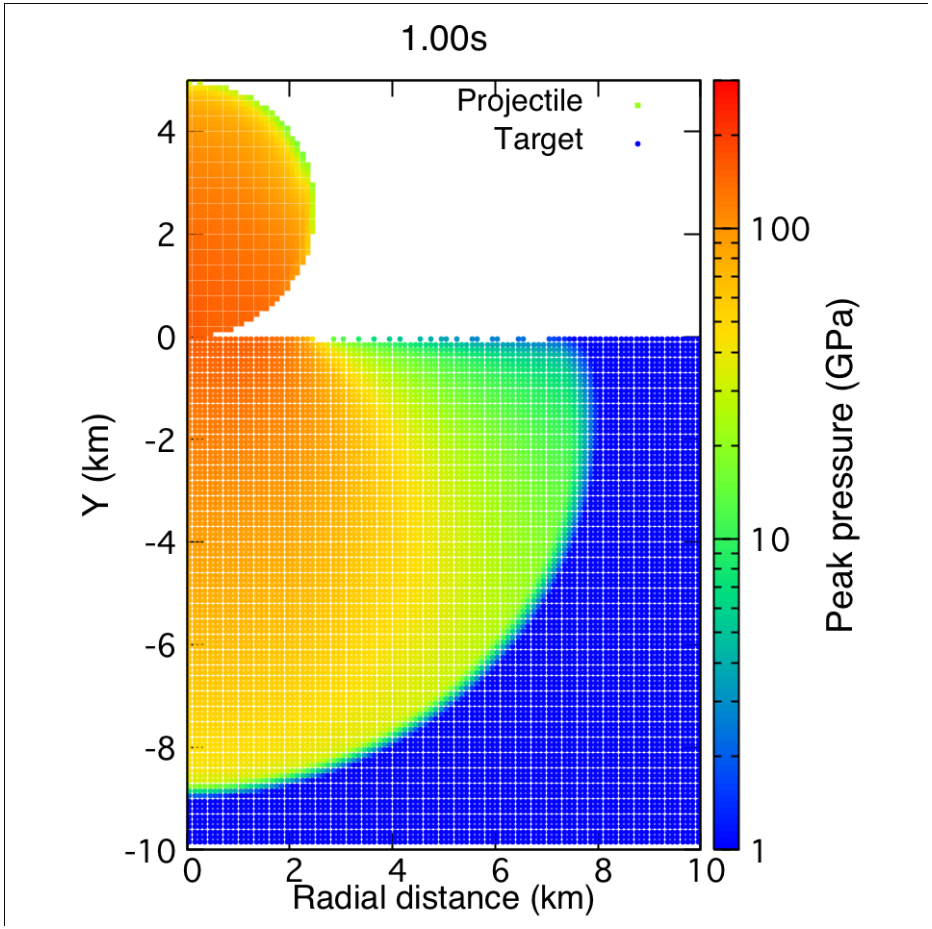
出力画像例

直径 5 km, 12 km/s, 垂直衝突

衝突後1 s後の圧力分布



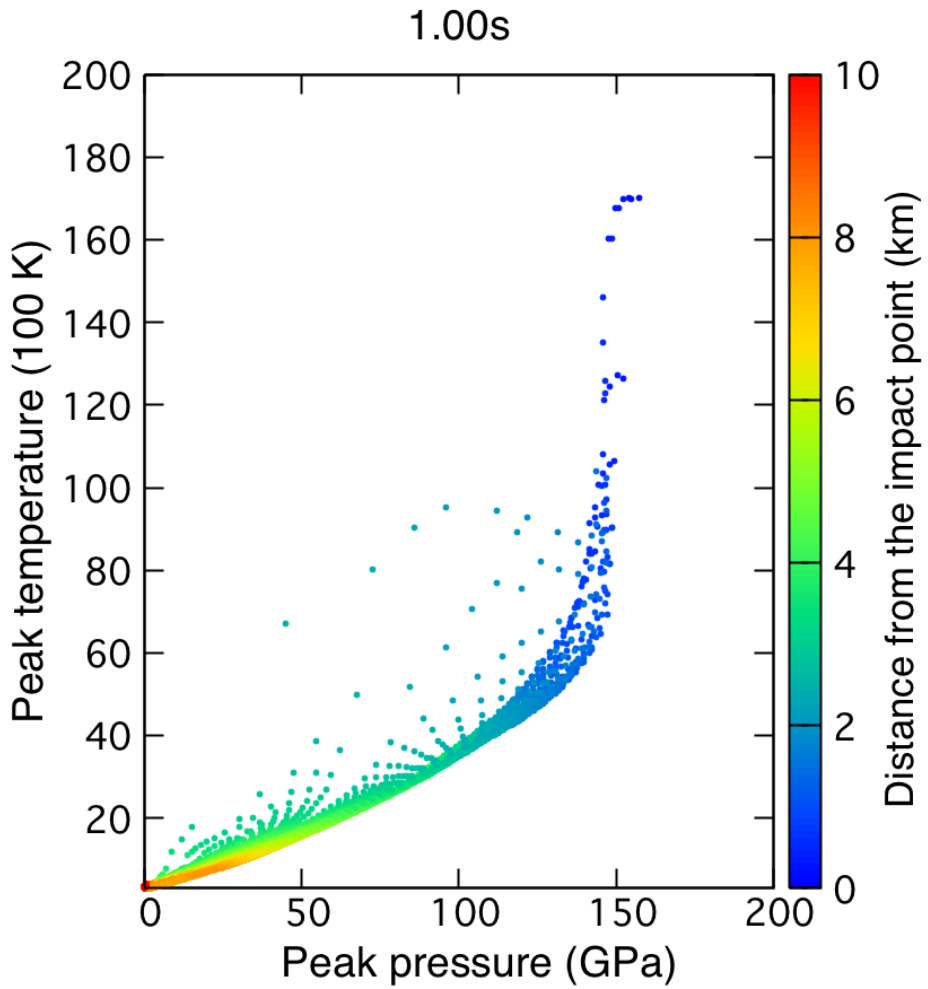
初期位置に対する最大衝撃圧分布



途中でエラーがなければ,
衝突後2秒までの5種類のグラフが0.1秒おき(計100枚)に格納.

出力画像例

直径 5 km, 12 km/s, 垂直衝突



最大衝撃圧 vs 最大温度
~Hugoniot曲線

サンプルプログラム
では何を行っているか？

計算の流れ

1. iSALE-2Dで計算を実施.
   「./Sample2D」フォルダの中にjdata.datが作成される.

2. iSALEPlotでトレーサデータ(X, Y, Pmax, Tmax)を書き出す.
   「./export」に格納される.「./export_max」に名称変更.

3. iSALEPlotでトレーサデータ(X, Y, P, T)を書き出す.
   「./export」に格納される.「./export_max」に名称変更.

4. Tracer_analysis_ex.cでトレーサ粒子データを読み込んで,
   解析. 結果は「./Processed/Data」に格納される.

5. Gnuplotで描画する.

iSALEPlotによるトレーサ粒子書き出し

「./Plotting/TrpTrt.inp」の中身（iSALEPlotのインプット）

```
#ISPLT
-----------------------------------------------------------------------
VERSION    __DO NOT MODIFY__                          : 2.0
--------- PLOT TYPES AND TIME STEPS ------------------------------------
PLOTTYPE   Type of plot (right and left panel         : Trp : Trt
TIMESTEP   First and last files to be read            :   0 :  200
INCREMNT   Spacing between files to be read           :   1
--------- GRID GEOMETRY ------------------------------------------------
GRIDH      Min and max i number for plotting          :   1 :   1
GRIDV      Min and max j number for plotting          :   1 :   1
--------- TRACER PARAMETERS --------------------------------------------
TR_TYPE    Tracer lines, grid or points     (1,2,3)   :   1
TR_SPACE   Spacing between tracers to be plotted      :   10
TR_SIZE    Size of tracer plot marker                 :   1
--------- PLOTTING PARAMETERS ------------------------------------------
TITLES     Should titles be plotted?  (1=yes,0=no)    :   1
PANEL      Double (2) or single (1) panel plot        :   -2
PLOTSIZE   Plot width and Height (inches)             :   12. :  9.
DEVICE     Graphics device and extension              :   png/png
```

ポイントは赤字で囲んだ部分. 出力は次ページ

iSALEPlotによるトレーサ粒子出力例

```
#Data exported at simulation time:   0.1000E+01
#Data columns: tracer unit, tracer IDTrp, Trt
#Total number of tracer units:       2
 #Number of tracers per unit:      988    9900
   1          1  1.2247E+02 -5.7355E+03  1.1097E+10  3.6122E+03
   1          2  1.1680E+02 -5.7230E+03  1.1097E+10  3.6122E+03
   1          3  1.1077E+02 -5.7092E+03  1.1097E+10  3.6122E+03
   1          4  1.0405E+02 -5.6927E+03  1.0128E+10  3.8999E+03
   1          5  9.7566E+01 -5.6741E+03  1.0086E+10  3.9126E+03
   1          6  9.1850E+01 -5.6530E+03  1.0086E+10  3.9126E+03
   1          7  8.6943E+01 -5.6269E+03  1.0086E+10  3.9126E+03
   1          8  8.2886E+01 -5.5938E+03  8.9183E+09  3.8625E+03
   1          9  7.9716E+01 -5.5544E+03  8.9183E+09  3.8625E+03
   1         10  7.7365E+01 -5.5120E+03  8.9183E+09  3.8625E+03
   1         11  7.5682E+01 -5.4689E+03  7.8867E+09  3.7965E+03
   1         12  7.4546E+01 -5.4250E+03  7.8867E+09  3.7965E+03
   1         13  7.3898E+01 -5.3796E+03  7.1330E+09  3.7335E+03
   1         14  7.3712E+01 -5.3329E+03  7.1330E+09  3.7335E+03
   1         15  7.3942E+01 -5.2857E+03  6.4499E+09  3.6591E+03
   1         16  7.4516E+01 -5.2389E+03  6.4499E+09  3.6591E+03
   1         17  7.5371E+01 -5.1930E+03  5.9063E+09  3.5795E+03
   1         18  7.6464E+01 -5.1480E+03  5.9063E+09  3.5795E+03
   1         19  7.7759E+01 -5.1042E+03  5.9063E+09  3.5795E+03
   1         20  7.9226E+01 -5.0618E+03  5.3519E+09  3.4849E+03
   1         21  8.0833E+01 -5.0208E+03  5.3519E+09  3.4849E+03
   1         22  8.2560E+01 -4.9812E+03  4.7048E+09  3.3714E+03
   1         23  8.4379E+01 -4.9432E+03  4.7048E+09  3.3714E+03
   1         24  8.6284E+01 -4.9067E+03  4.7048E+09  3.3714E+03
   1         25  8.8262E+01 -4.8716E+03  3.8886E+09  3.2387E+03
   1         26  9.0291E+01 -4.8380E+03  3.8886E+09  3.2387E+03
```

Index　通番号　*R*座標　*Z*座標　圧力　温度

一つのファイルにある時刻の
全トレーサ粒子の
*X, Y, P, T*が格納されている.

トレーサ粒子解析プログラムの流れ

1. 「./export」と「./export_max」からトレーサ粒子データ TrpTrtXXXXX.txtを読み込み, 配列に格納.

2. 適当な条件処理, もしくは演算を施し調べたい物理量を算出する.

3. データを書き出す.

今回のサンプルプログラムでは衝突点からの
距離をトレーサ粒子の*X*, *Y*座標から算出している.
同じ要領で様々な物理量を算出できる(はず).

トレーサ粒子解析プログラムの解説1

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

#define STR_MAX 256
#define PI M_PI
#define CHARACTER_NUM1 10
#define CHARACTER_NUM2 300

#define NUMBER1 20000 /*Number of tracer particles*/

int main(int argc , char *argv[]){

    /*変数定義*/

    int line;

    int i, j, k, l, m, n, o, q, p;
    int Time_stamp, Time_stamp_initial;
    int Proj_count, Target_count;
    int Thin_number_time, Thin_number_proj, Thin_number_target;

    int Tracer_number, *Tracer_number_proj, *Tracer_number_target;
    int Tracer_index, Tracer_index_proj, Tracer_index_target;

    double *X, *Y, *Trp, *Trt, *TrPmax, *TrTmax;
    double *X_proj, *Y_proj, *Trp_proj, *Trt_proj, *TrPmax_proj, *TrTmax_proj;
    double *X_target, *Y_target, *Trp_target, *Trt_target, *TrPmax_target, *TrTmax_target;

    double *X_proj0, *Y_proj0;
    double *X_target0, *Y_target0;

    double *R_proj, *R_proj0;
    double *R_target, *R_target0;

    double Time, Time_increment;
```

確保する配列の数
トレーサ粒子の数より多くしておく
必要あり.

変数の定義
・名前は変更自由
・トレーサ粒子に関する量
はポインタで定義して
おくこと.
*(こめ)をつける.

トレーサ粒子解析プログラムの解説2

```
    Tracer_number_proj = (int *)malloc(sizeof(int)*NUMBER1);
    Tracer_number_target = (int *)malloc(sizeof(int)*NUMBER1);

    X = (double *)malloc(sizeof(double)*NUMBER1);
    Y = (double *)malloc(sizeof(double)*NUMBER1);
    Trp = (double *)malloc(sizeof(double)*NUMBER1);
    Trt = (double *)malloc(sizeof(double)*NUMBER1);
    TrPmax = (double *)malloc(sizeof(double)*NUMBER1);
    TrTmax = (double *)malloc(sizeof(double)*NUMBER1);

    X_proj = (double *)malloc(sizeof(double)*NUMBER1);
    Y_proj = (double *)malloc(sizeof(double)*NUMBER1);
    X_proj0 = (double *)malloc(sizeof(double)*NUMBER1);
    Y_proj0 = (double *)malloc(sizeof(double)*NUMBER1);

    R_proj = (double *)malloc(sizeof(double)*NUMBER1);
    R_proj0 = (double *)malloc(sizeof(double)*NUMBER1);
    R_target = (double *)malloc(sizeof(double)*NUMBER1);
    R_target0 = (double *)malloc(sizeof(double)*NUMBER1);

    Trp_proj = (double *)malloc(sizeof(double)*NUMBER1);
    Trt_proj = (double *)malloc(sizeof(double)*NUMBER1);
    TrPmax_proj = (double *)malloc(sizeof(double)*NUMBER1);
    TrTmax_proj = (double *)malloc(sizeof(double)*NUMBER1);

    X_target = (double *)malloc(sizeof(double)*NUMBER1);
    Y_target = (double *)malloc(sizeof(double)*NUMBER1);
    X_target0 = (double *)malloc(sizeof(double)*NUMBER1);
    Y_target0 = (double *)malloc(sizeof(double)*NUMBER1);
    Trp_target = (double *)malloc(sizeof(double)*NUMBER1);
    Trt_target = (double *)malloc(sizeof(double)*NUMBER1);
    TrPmax_target = (double *)malloc(sizeof(double)*NUMBER1);
    TrTmax_target = (double *)malloc(sizeof(double)*NUMBER1);
```

メモリ確保

・新しいポインタ変数を定義したら
同じ要領で追加する.

トレーサ粒子解析プログラムの解説3

```
114
115     /*変数の初期化*/
116     for(i=0; i < NUMBER1; i++) Tracer_number_proj[i] = 0;
117     for(i=0; i < NUMBER1; i++) Tracer_number_target[i] = 0;
118
119     for(i=0; i < NUMBER1; i++) X_proj[i] = 0.;
120     for(i=0; i < NUMBER1; i++) Y_proj[i] = 0.;
121     for(i=0; i < NUMBER1; i++) X_proj0[i] = 0.;
122     for(i=0; i < NUMBER1; i++) Y_proj0[i] = 0.;
123     for(i=0; i < NUMBER1; i++) R_proj[i] = 0.;
124     for(i=0; i < NUMBER1; i++) R_proj0[i] = 0.;
125     for(i=0; i < NUMBER1; i++) Trp_proj[i] = 0.;
126     for(i=0; i < NUMBER1; i++) Trt_proj[i] = 0.;
127     for(i=0; i < NUMBER1; i++) TrPmax_proj[i] = 0.;
128     for(i=0; i < NUMBER1; i++) TrTmax_proj[i] = 0.;
129
130     for(i=0; i < NUMBER1; i++) X_target[i] = 0.;
131     for(i=0; i < NUMBER1; i++) Y_target[i] = 0.;
132     for(i=0; i < NUMBER1; i++) X_target0[i] = 0.;
133     for(i=0; i < NUMBER1; i++) Y_target0[i] = 0.;
134     for(i=0; i < NUMBER1; i++) R_target[i] = 0.;
135     for(i=0; i < NUMBER1; i++) R_target0[i] = 0.;
136     for(i=0; i < NUMBER1; i++) Trp_target[i] = 0.;
137     for(i=0; i < NUMBER1; i++) Trt_target[i] = 0.;
138     for(i=0; i < NUMBER1; i++) TrPmax_target[i] = 0.;
139     for(i=0; i < NUMBER1; i++) TrTmax_target[i] = 0.;
```

配列の初期化

・新しいポインタ変数を定義したら同じ要領で追加する.

トレーサ粒子解析プログラムの解説4

```
    m = 0;    /*Projectile内のトレーサ粒子を数えるための整数*/
    n = 0;    /*Target内    のトレーサ粒子を数えるための整数*/
    line = 0; /*TrpTrtXX.txtの行数を        数えるための整数*/

    while(fgets(buff, sizeof(buff), fpr_in1) != NULL){    /*ファイルの中から1行づつ読み込む*/
        ++line;    /* 読み込んだ行数をカウントアップ*/

        if(line <= 4) continue; /*ヘッダの４行はスキップする*/

        sscanf(buff, "%d %d %lf %lf %lf %lf", &Tracer_index, &Tracer_number, &X[k], &Y[k], &Trp[k], &Trt[k]);

        if(Tracer_index == Tracer_index_proj){

            Tracer_number_proj[m] = Tracer_number;
            X_proj[m]   = X[k];
            Y_proj[m]   = Y[k];
            Trp_proj[m] = Trp[k];
            Trt_proj[m] = Trt[k];

            R_proj[m] = sqrt(X_proj[m]*X_proj[m] + Y_proj[m]*Y_proj[m]);

            if(k == Time_stamp_initial){

                Trt_proj[m] = Temp0;
                X_proj0[m]   = X[k];
                Y_proj0[m]   = Y[k];

                R_proj0[m] = sqrt(X_proj0[m]*X_proj0[m] + Y_proj0[m]*Y_proj0[m]);

            }

            if(Trp_proj[m] < Pressure_min) Trp_proj[m] = Pressure_min;

            m++;

        }
```

C言語のfgets関数とsscanf関数で
トレーサ粒子データを読み込む

座標データの演算で
衝突点からの距離を算出

時刻0におけるのときの
座標を記録

トレーサ粒子解析プログラムの解説5

```
291         if(k%Thin_number_time==0){
292
293             sprintf(file_proj, "./Processed/Data/TrpTrt_projectile_%1.2fs.txt", Time);
294             sprintf(file_target, "./Processed/Data/TrpTrt_target_%1.2fs.txt", Time);
295
296             fp_proj = fopen(file_proj,"w");
297
298             fprintf(fp_proj,"%1s", "#Tracer_number");
299             fprintf(fp_proj," %1s","Time[s]");
300             fprintf(fp_proj," %1s %1s","X0[m]", "Y0[m]");
301             fprintf(fp_proj," %1s %1s","X[m]", "Y[m]");
302             fprintf(fp_proj," %1s %1s","Pressure[GPa]", "log10(Pressure)");
303             fprintf(fp_proj," %1s","Temperature[K]");
304             fprintf(fp_proj," %1s %1s","Pmax[GPa]", "log10(Pmax[GPa])");
305             fprintf(fp_proj," %1s","Tmax[K]");
306             fprintf(fp_proj," %1s %1s","Distance0[m]", "Distance[m]");
307             fprintf(fp_proj,"\n");
```

データ書き出し

・Projectile(index=1)がTrpTrt_projectile_XXXs.txtに
Target(index=2)　がTrpTrt_target_XXXs.txtにそれぞれ書き出される.
->各タイムステップで2つのファイルが生成される.

・各出力ファイルの一行目に何を出力するかを名付ける.
名称は自由に変更可能だがスペースは入れないこと. (空白区切りのデータ)
例. Pressure[GPa] -> P[GPa]はOK, P [GPa]はダメ.

トレーサ粒子解析プログラムの解説6

```
322        for(m=0; m<Proj_count/Thin_number_proj; m++){
323
324            int p = m*Thin_number_proj;
325
326            if(Temp_min < Trt_proj[p] && Trt_proj[p] < Temp_max && Temp_min < TrTmax_proj[p] && TrTmax_proj[p] < Temp_max){
327
328                fprintf(fp_proj,"%1.4d", Tracer_number_proj[p]);
329                fprintf(fp_proj," %1.4e",Time);
330                fprintf(fp_proj," %1.4e %1.4e",X_proj0[p], Y_proj0[p]);
331                fprintf(fp_proj," %1.4e %1.4e",X_proj[p], Y_proj[p]);
332                fprintf(fp_proj," %1.4e %1.4e",Trp_proj[p]*1.e-9, log10(Trp_proj[p]));
333                fprintf(fp_proj," %1.4e",Trt_proj[p]);
334                fprintf(fp_proj," %1.4e %1.4e",TrPmax_proj[p]*1.e-9, log10(TrPmax_proj[p]));
335                fprintf(fp_proj," %1.4e",TrTmax_proj[p]);
336                fprintf(fp_proj," %1.4e %1.4e", R_proj0[p], R_proj[p]);
337                fprintf(fp_proj,"\n");
338
339            }
340
341        }
```

解析結果を出力.
変数の順番は, 全ページでつけた名前順にしなければならないことに注意.

計算出力に対するフィルター
例えば, ある10 GPaを超えたトレーサ粒子だけを出力することも可能.
Ex. TrPmax_proj[p] > 10.e+9, をif()に追加すればよい.

トレーサ粒子解析プログラムの解説7

```
    free(X);
    free(Y);
    free(Trp);
    free(Trt);
    free(TrPmax);
    free(TrTmax);

    free(X_proj);
    free(Y_proj);
    free(X_proj0);
    free(Y_proj0);
    free(R_proj);
    free(R_proj0);
    free(Trp_proj);
    free(Trt_proj);
    free(TrPmax_proj);
    free(TrTmax_proj);

    free(X_target);
    free(Y_target);
    free(X_target0);
    free(Y_target0);
    free(R_target);
    free(R_target0);
    free(Trp_target);
    free(Trt_target);
    free(TrPmax_target);
    free(TrTmax_target);
```

メモリの開放.
新しくポインタ変数を追加した場合はここにも追加しておくこと.

シェルスクリプトの解説1

シェルスクリプト: 処理が終わるのを待って, 上から順にターミナルに自動入力してくれる.

```
./iSALE2D   <- iSALE-2Dを実行

./iSALEPlot -f ./Plotting/TrPTrT_max.inp -m ./Sample2D/jdata.dat
mv export export_max
./iSALEPlot -f ./Plotting/TrpTrt.inp -m ./Sample2D/jdata.dat

mkdir Processed
mkdir Processed/Data
mkdir Processed/Figures_png
mkdir Processed/Figures_eps
```

iSALEPlotによるTracer粒子書き出し

C言語の解析プログラム&gnuplotの出力を格納するためのディレクトリを生成

シェルスクリプトの解説2

```
17 cc -O3 -o Tracer_Analysis_ex Tracer_Analysis_ex.c -lm
18 ./Tracer_Analysis_ex
```

解析プログラムをコンパイル&実行

※解析プログラムを改良した直後はコンパイルコマンドを打ち込んで,
コンパイルが通ることを確認したほうがよい.

シェルスクリプトの解説3

```
for j in 0 1

    do

    for i in 0 1 2 3 4 5 6 7 8 9

    do

        echo Making figures at "$j"."$i"0s after the impact.
        sed "s/1.00s/"$j"."$i"0s/g" Tracer_plot_ex.plt | gnuplot
        sed "s/1.00s/"$j"."$i"0s/g" Pmax-Tmax-Ro.plt | gnuplot

    done #i

done #j
```

２つのgnuplotスクリプトを時間順に実行.

gnuplotによる描画.

- Tracer_plot_ex.plt
- Pmax-Tmax-Ro.plt

sedコマンドでファイル内の1.00sを任意の時間に書き換えてgnuplotに渡している.

シェルスクリプトの解説4

解析をやり直す場合, コメントアウトを活用すべし.

```
3  : << '#_comment_out'
4  #_comment_out
5
6  ./iSALE2D
7
8  ./iSALEPlot -f ./Plotting/TrPTrT_max.inp -m ./Sample2D/jdata.dat
9  mv export export_max
10 ./iSALEPlot -f ./Plotting/TrpTrt.inp -m ./Sample2D/jdata.dat
11
12 mkdir Processed
13 mkdir Processed/Data
14 mkdir Processed/Figures_png
15 mkdir Processed/Figures_eps
16
17 cc -O3 -o Tracer_Analysis_ex Tracer_Analysis_ex.c -lm
18 ./Tracer_Analysis_ex
19
```

C言語のプログラムを書き換えた後に
再度iSALE-2Dとトレーサ書き出しを実行する必要は
必ずしもない.

#_comment_outをカットアンドペーストで移動させると, 2つのコメントアウトで挟んでいる
行をまとめてコメントアウトできる.