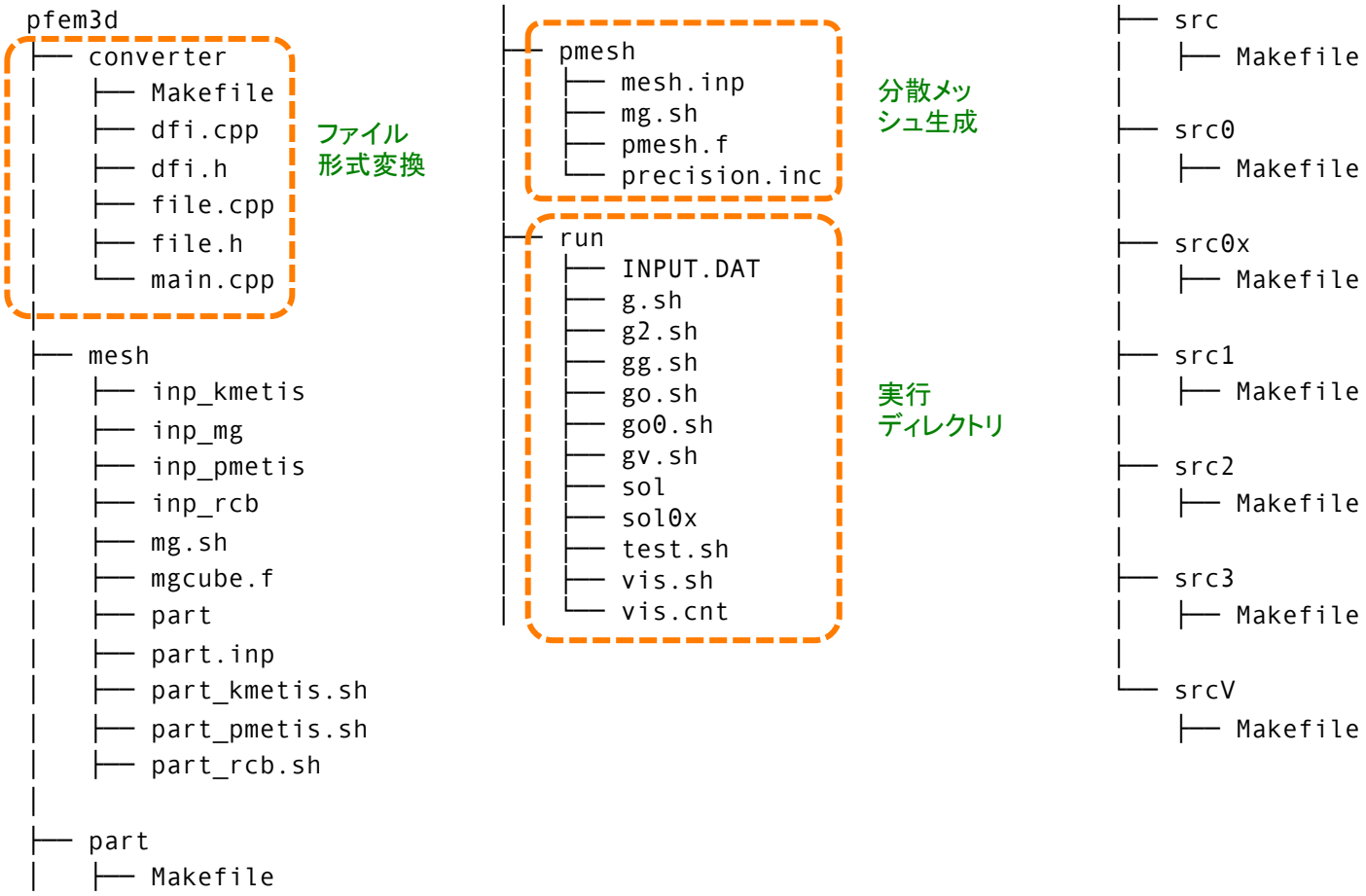
Hands-on

小野 謙二
理化学研究所計算科学研究機構

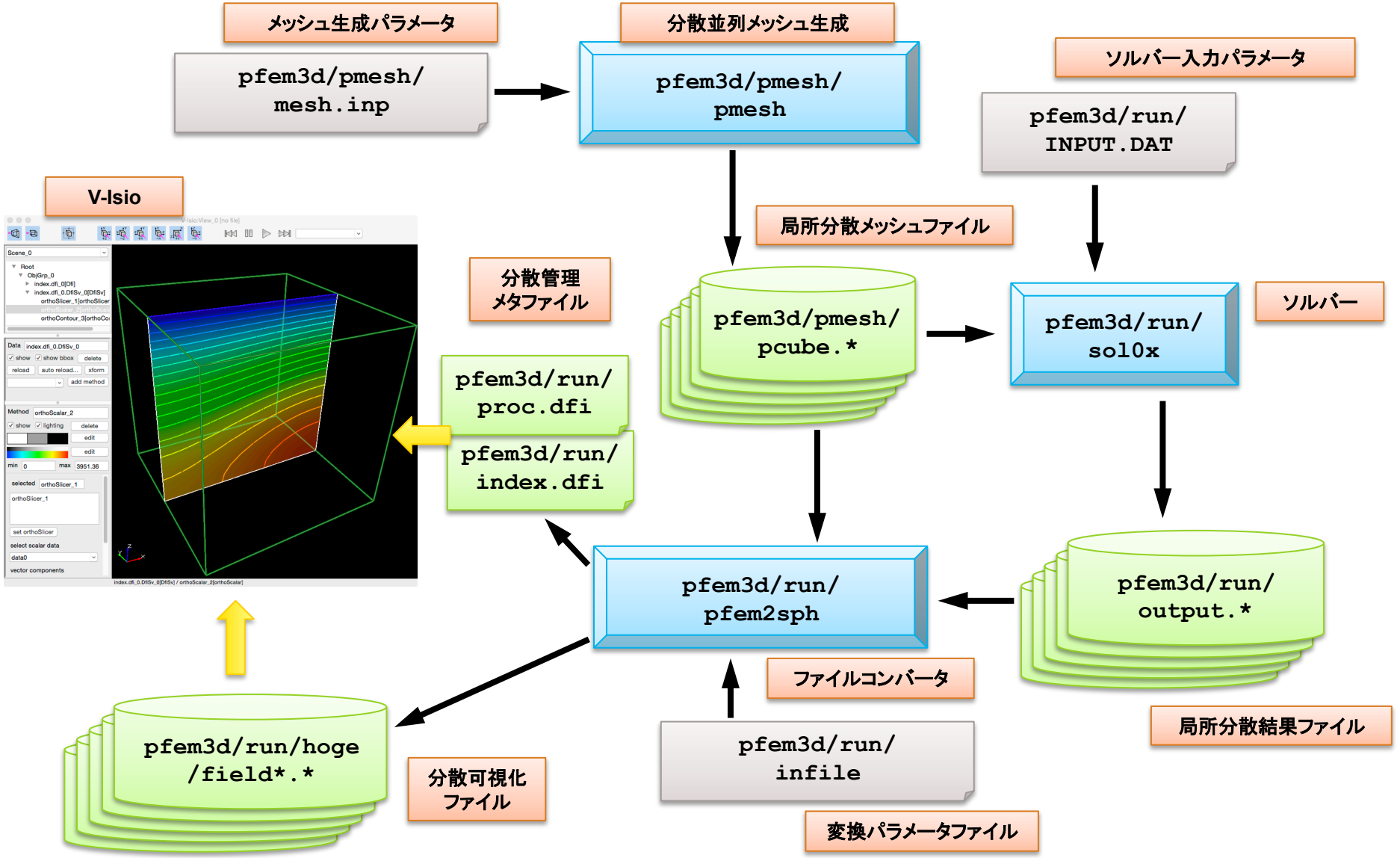
体験すること

1. 並列で分散メッシュを生成 pmesh
2. ソルバーのコンパイル sol0x
3. 並列有限要素法で並列計算実行 sol0x
4. 計算結果を変換 pfem2sph
5. 結果をPCに転送 index.dfi, proc.dfi, output.*
6. V-Isioで可視化
   - 断面分布
   - 断面等高線
   - 等値面
   - ボリュームレンダリング

初期ディレクトリ構造と主要ファイル



並列有限要素法と可視化の手順



1. 並列メッシュ生成

スライドファイル 07-pFEM3D-1.ppt, p.49

- コンパイル

```
>$ pwd
/home/S11502/keno/pFEM/pfem3d/pmesh

>$ mpifrtpx -Kfast pmesh.f -o pmesh
```

- 分割パラメータ

```
>$ cat mesh.inp
20 20 20   // 軸方向の節点数
2 2 2      // 軸方向の分割数
pcube      // 分割ファイル名のプリフィクス
```

割り切れること

- ジョブ実行

```
>$ pjsub mg.sh
>$ pjstat
>$ ls
… pcube.0  pcube.1  pcube.2  pcube.3
pcube.4  pcube.5  pcube.6  pcube.7 …
```

Flat MPIで8並列の場合

ジョブスクリプト

```
>$ cat mg.sh
#!/bin/sh
#PJM -L "node=1"
#PJM -L "elapse=00:05:00"
#PJM -L "rscgrp=small"
#PJM -j
#PJM -o "mg.lst"
#PJM --mpi "proc=8"

mpiexec ./pmesh
rm wk.*
```

2x2x2=8

2. ソルバーのコンパイル

```
>$ cd src0x
>$ cat Makefile

F90      =  mpifrtpx
F90LINKER = $(F90)
LIB_DIR   =
INC_DIR   =
OPTFLAGS  = -Kfast,ocl
FFLAGS = $(OPTFLAGS)
FLIBS =
F90LFLAGS=
#
TARGET = ../run/sol0x
default: $(TARGET)
OBJS =\
pfem_util.o \
solver_SR.o solver_CG.o \
solver11.o test1.o util.o pfem_init.o input_cntl.o input_grid.o define_file_name.o\
mat_con0.o mat_con1.o mat_ass_main.o mat_ass_bc.o pfem_finalize.o

$(TARGET):  $(OBJS)
      $(F90LINKER) $(OPTFLAGS) -o $(TARGET) $(OBJS) $(F90LFLAGS)
clean:
      /bin/rm -f *.o $(TARGET) *~ *.mod
.f.o:
      $(F90) $(FFLAGS) $(INC_DIR) -c  $*.f
.f90.o:
      $(F90) $(FFLAGS) $(INC_DIR) -c  $*.f90
.SUFFIXES: .f90 .f
```

コンパイルの実行

```
>$ make
```

3. ソルバーの実行

ジョブスクリプト

```
>$ cat test.sh
#!/bin/sh
#PJM -L "node=1"
#PJM -L "elapse=00:10:00"
#PJM -L "rscgrp=small"
#PJM -j
#PJM -o "test.lst"
#PJM --mpi "proc=8"

mpiexec ./sol0x
```

入力パラメータファイル

```
>$ cat INPUT.DAT
../pmesh/pcube
2000
1.0 1.0
1.0e-08
```

ジョブ実行

```
>$ pjsub test.sh
>$ pjstat
```

3. ソルバーの実行

ジョブ実行後

```
>$ ls -la
drwxr-xr-x  3 keno S11502    4096  8月 15 18:06 2015 .
drwxr-xr-x 14 keno S11502    4096  8月 15 17:50 2015 ..
-rwxr-xr-x  1 keno S11502      36  5月 14 10:20 2014 INPUT.DAT
-rw-r--r--  1 keno S11502   35958  8月 10 08:35 2015 output.0
-rw-r--r--  1 keno S11502   35958  8月 10 08:35 2015 output.1
-rw-r--r--  1 keno S11502   35958  8月 10 08:35 2015 output.2
-rw-r--r--  1 keno S11502   35958  8月 10 08:35 2015 output.3
-rw-r--r--  1 keno S11502   35958  8月 10 08:35 2015 output.4
-rw-r--r--  1 keno S11502   35958  8月 10 08:35 2015 output.5
-rw-r--r--  1 keno S11502   35958  8月 10 08:35 2015 output.6
-rw-r--r--  1 keno S11502   35958  8月 10 08:35 2015 output.7
```

output.? ファイルがプロセス数だけ生成される

実行結果

```
$> cat test.lst
*** matrix conn.     5.959041E-03 sec.
*** matrix ass.      6.912349E-03 sec.

    1    3.904523E+00
    2    3.504358E+00
    3    3.196968E+00
    4    2.954789E+00
    …
    …
    …
   55    1.034594E-07
   56    3.225752E-08
   57    1.114609E-08
   58    5.284856E-09
*** real  COMP.      2.967281E-02 sec.

jwe0002i stop  * normal termination
```

4. 計算結果の変換

```
>$ pwd
/home/S11502/keno/pFEM/pfem3d/converter

>$ cat Makefile
RM          = \rm -f
CXX         = FCCpx
CXXFLAGS    = -Kfast,ocl
LDFLAGS     =
TARGET      = ../run/pfem2sph

default: $(TARGET)
SRCS = main.cpp file.cpp dfi.cpp
.SUFFIXES: .o .cpp
OBJS = $(SRCS:.cpp=.o)

$(TARGET): $(OBJS)
      $(CXX) $(CXXFLAGS) -o $(@) $(OBJS) $(LDFLAGS)
.cpp.o:
      $(CXX) $(CXXFLAGS) -c $<
clean:
      $(RM) $(OBJS) pfem2sph
```

逐次でコンパイル、実行

コンパイルの実行

```
>$ make
```

4. 計算結果の変換

ジョブスクリプト

```
>$ cat vis.sh
#!/bin/sh
#PJM -L "node=1"
#PJM -L "elapse=00:10:00"
#PJM -L "rscgrp=small"
#PJM -j
#PJM -o "vis.lst"
#PJM --mpi "proc=1"

./pfem2sph infile
```

入力パラメータファイル

```
>$ cat infile
20 20 20          // mesh.inpの記述と同じ
2 2 2             //
../pmesh/pcube    // 相対パス
output            // 結果ファイル名のプリフィクス
./hoge            // 変換ファイルの出力先
```

hogeディレクトリは先に作っておく
>$ mkdir hoge

ジョブ実行

```
>$ pjsub vis.sh
>$ ls hoge
field_0000000000_id000000.sph  field_0000000000_id000002.sph  field_0000000000_id000004.sph
field_0000000000_id000006.sph  field_0000000000_id000001.sph  field_0000000000_id000003.sph
field_0000000000_id000005.sph  field_0000000000_id000007.sph
```

今回は、タイムステップ=0に固定

生成メタファイル

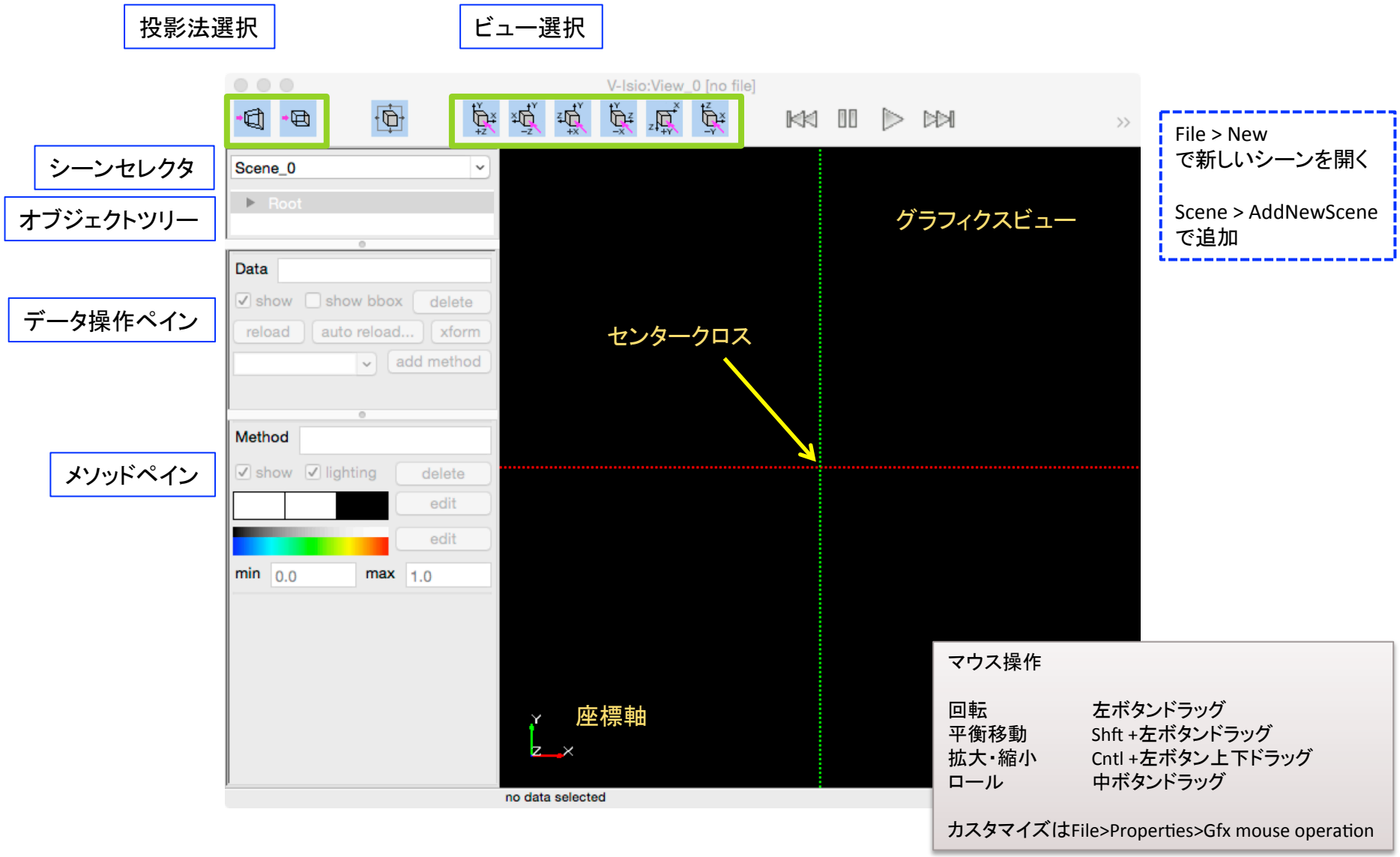
```
>$ cat index.dfi

FileInfo {
  DFIType             = "Cartesian"
  DirectoryPath       = "./hoge"
  TimeSliceDirectory = "off"
  Prefix              = "field"
  FileFormat          = "sph"
  FieldFilenameFormat= "step_rank"
  GuideCell           = 0
  DataType            = "Float32"
  Endian              = "little"
  NumVariables        = 1
  Variable[@]{ name   = "temperature" }
}

FilePath {
  Process = "proc.dfi”
}

TimeSlice {
  Slice[@] {
    Step = 0
    Time = 0.000000e+00
    MinMax[@] {
      Min = 0.000000e+00
      Max = 2.918599e+03
    }
  }
}
```

```
>$ cat proc.dfi

Domain {
  GlobalOrigin        = (0.0000e+00, 0.0000e+00, 0.0000e+00)
  GlobalRegion        = (1.9000e+01, 1.9000e+01, 1.9000e+01)
  GlobalVoxel         = (20, 20, 20)
  GlobalDivision      = (2, 2, 2)
  ActiveSubdomainFile = ""
}

MPI {
  NumberOfRank   = 8
  NumberOfGroup  = 1
}

Process {
  Rank[@] {
    ID        = 0
    HostName  = ""
    VoxelSize = (10, 10, 10)
    HeadIndex = (1, 1, 1)
    TailIndex = (10, 10, 10)
    CellID    = 1
    BCflagID  = 1
  }

  ...

}
```

V-Isioによる可視化

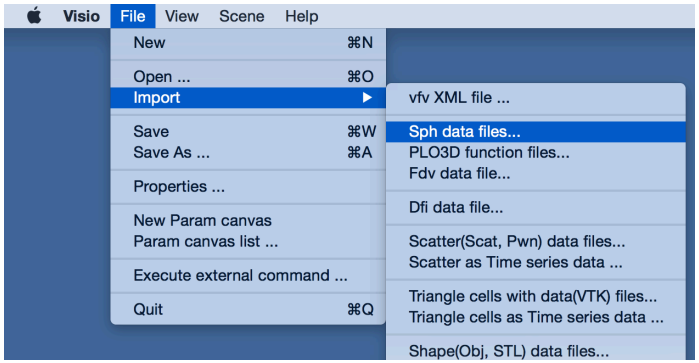
- V-Isio
  - 理研VCADプロジェクトで開発した可視化アプリ
  - 構造格子とメッシュデータを対象
  - 中規模程度のデータを可視化
    - 搭載メモリ依存であるが、スカラデータの場合、16GBで8億点程度まで可視化可能
  - マルチプラットホーム（Linux, Win, Mac）
  - http://avr-aics-riken.github.io/V-Isio/

Look & feel

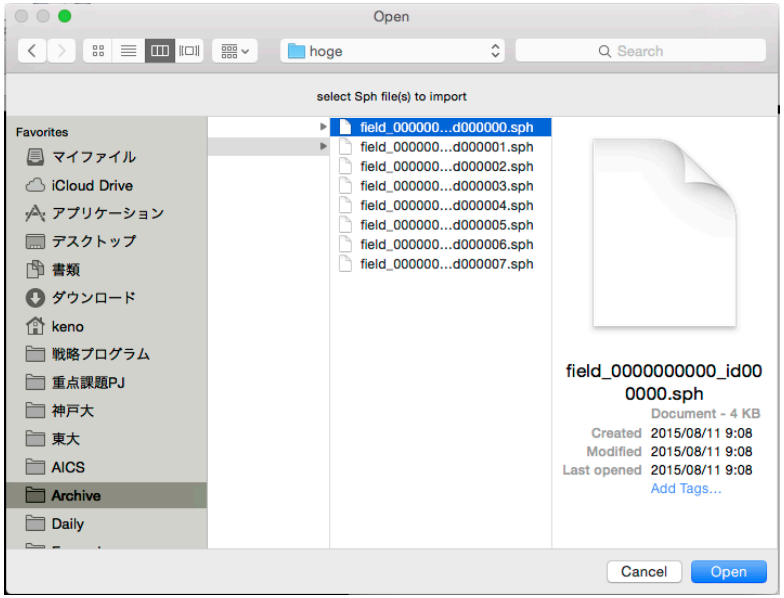

投影法選択
ビュー選択
シーンセレクタ
オブジェクトツリー
データ操作ペイン
メソッドペイン
V-Isio:View_0 [no file]
Scene_0
Root
Data
show
show bbox
delete
reload
auto reload...
xform
add method
Method
show
lighting
delete
edit
edit
min
0.0
max
1.0
グラフィクスビュー
センタークロス
座標軸
no data selected
File > New
で新しいシーンを開く
Scene > AddNewScene
で追加
マウス操作
回転 左ボタンドラッグ
平衡移動 Shft +左ボタンドラッグ
拡大・縮小 Cntl +左ボタン上下ドラッグ
ロール 中ボタンドラッグ
カスタマイズはFile>Properties>Gfx mouse operation

データロード

1. Sph data filesを選択

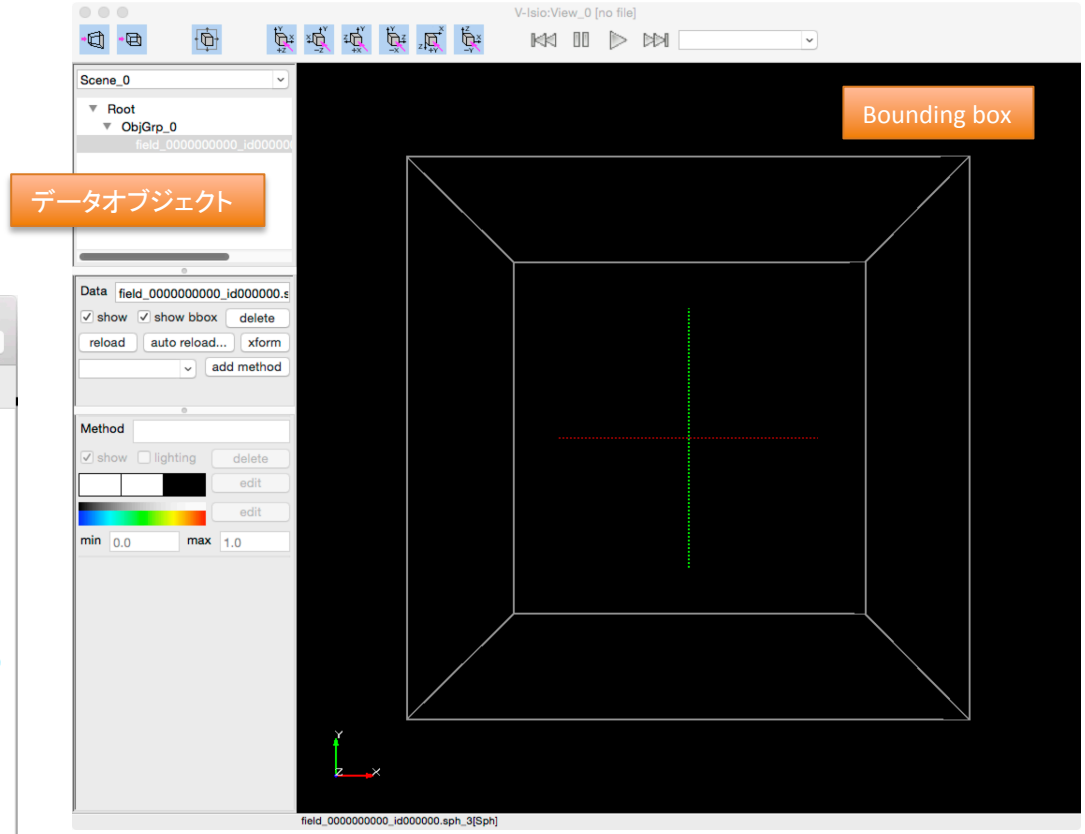


2. ロードするファイルを選択



3. sphファイルがロードされた状態

オブジェクトツリーに、ロード対象のファイル名がインポートされている。データロード直後の状態では、ロードされたオブジェクトが選択されており、データ操作ペインに対象のデータ名が表示されている。一つのシーンに複数のデータをロード可能。

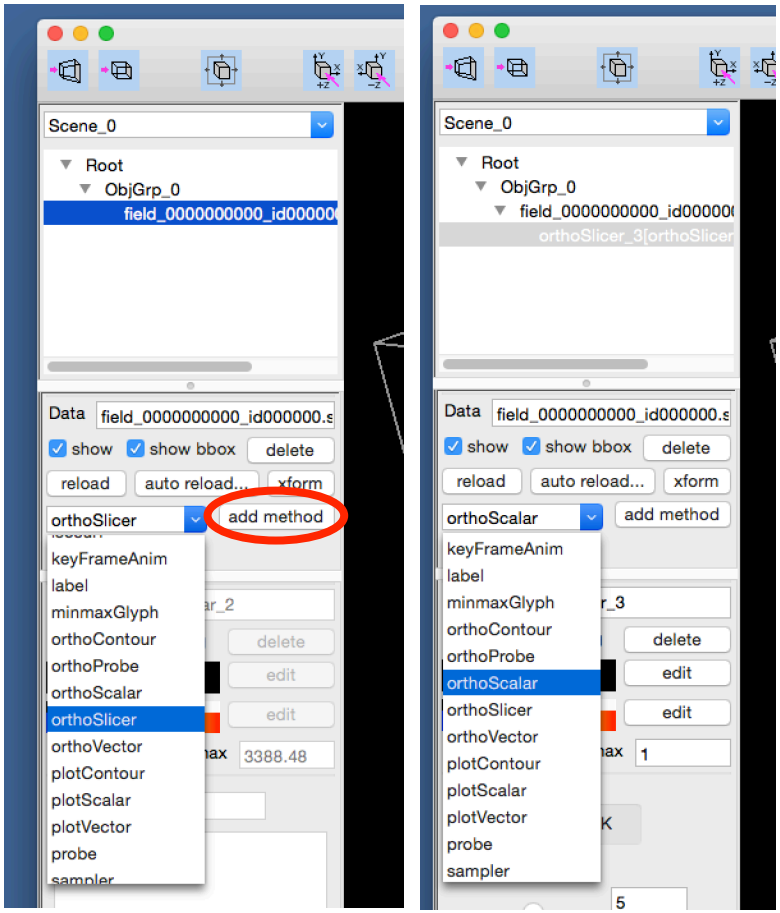


断面分布の表示

- スカラーデータを対象に、X/Y/Z各軸方向の断面で値の分布を表示する。
  - データ操作ペインのプルダウンメニューから、インポートしたデータに適用する可視化メソッドを選択する。ここでは、*ORTHO_SCALARを選択。選択後に、add methodをクリックする。

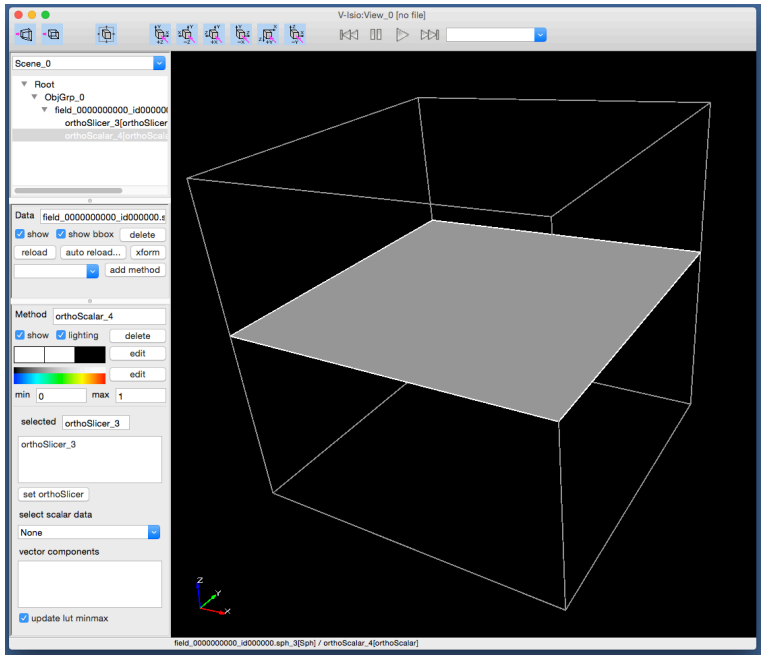
1. 断面指定のメソッドを追加
   >> orthoSlicer

2. スカラ断面分布表示のメソッドを追加
   >> orthoScalar

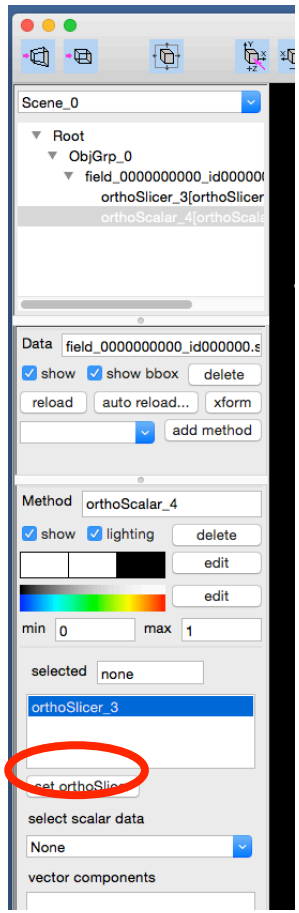


orthoScalarメソッド

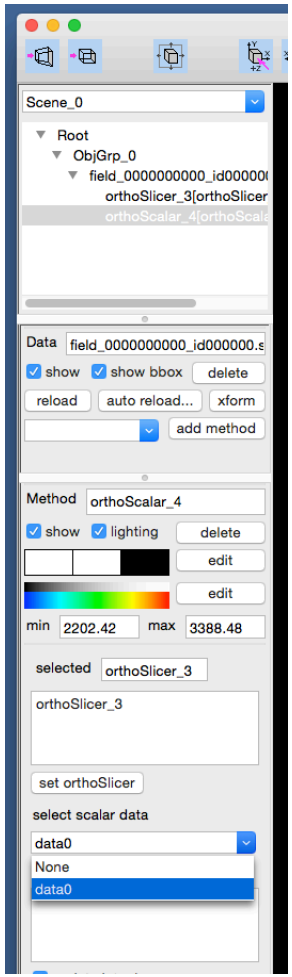
orthoSlicer, orthoScalarをデータに追加した状態



オブジェクトツリーでorthoScalarを選択し、メソッドペインで断面操作メソッドであるorthoSlicerを選択する。選択後、set orthoSlicerをクリック。



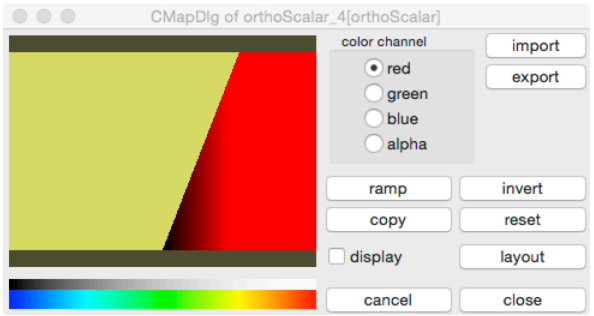
次に、メソッドペインで描画対象データを選択する。プルダウンメニューからdata0を選択。

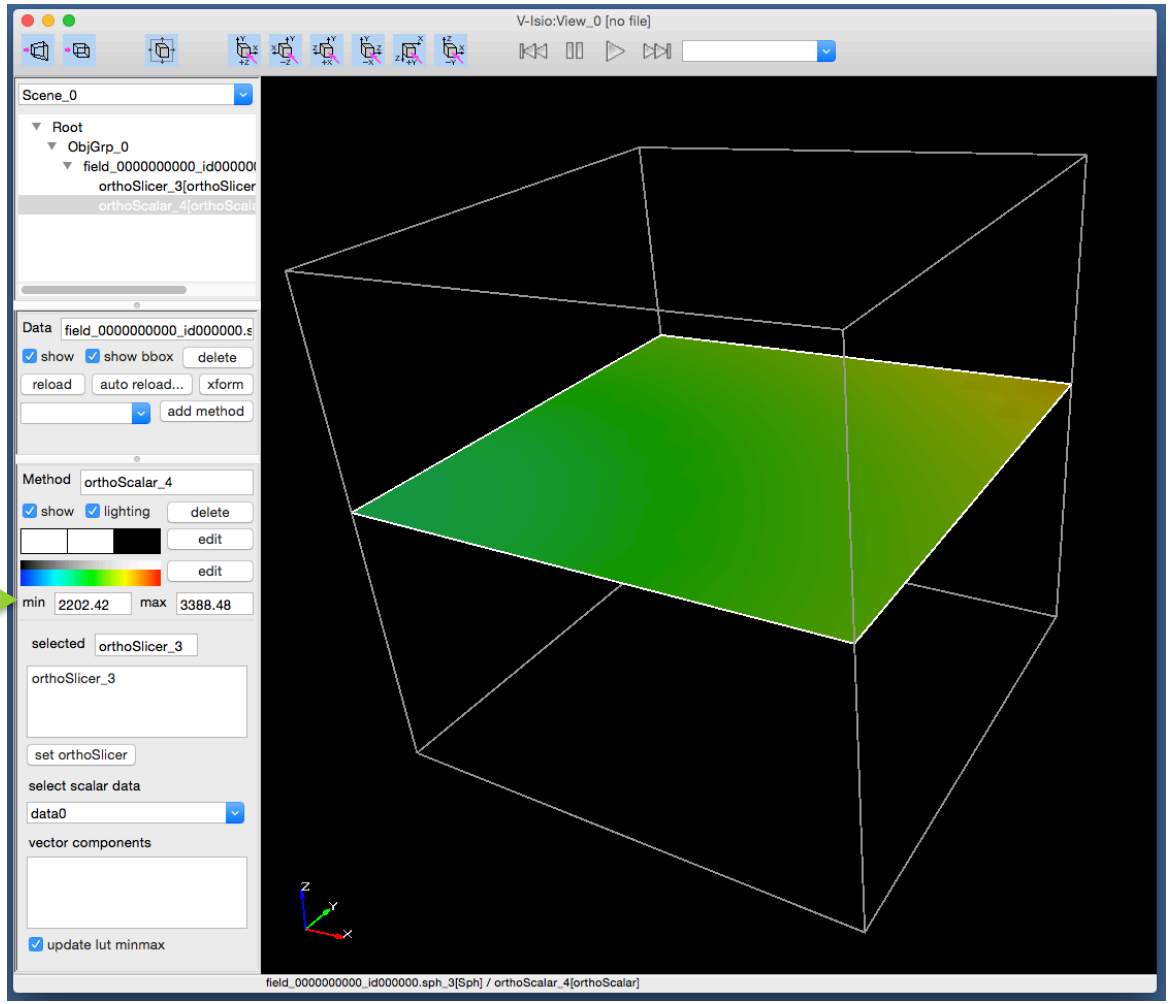


orthoScalarメソッド

データ field_0000000000_id000000.sph の最小値と最大値を表示している。最小値と最大値がカラーバーの両端の色に対応している。

カラーバー右横のeditボタンでカラーバーの編集ウインドウが現れる。



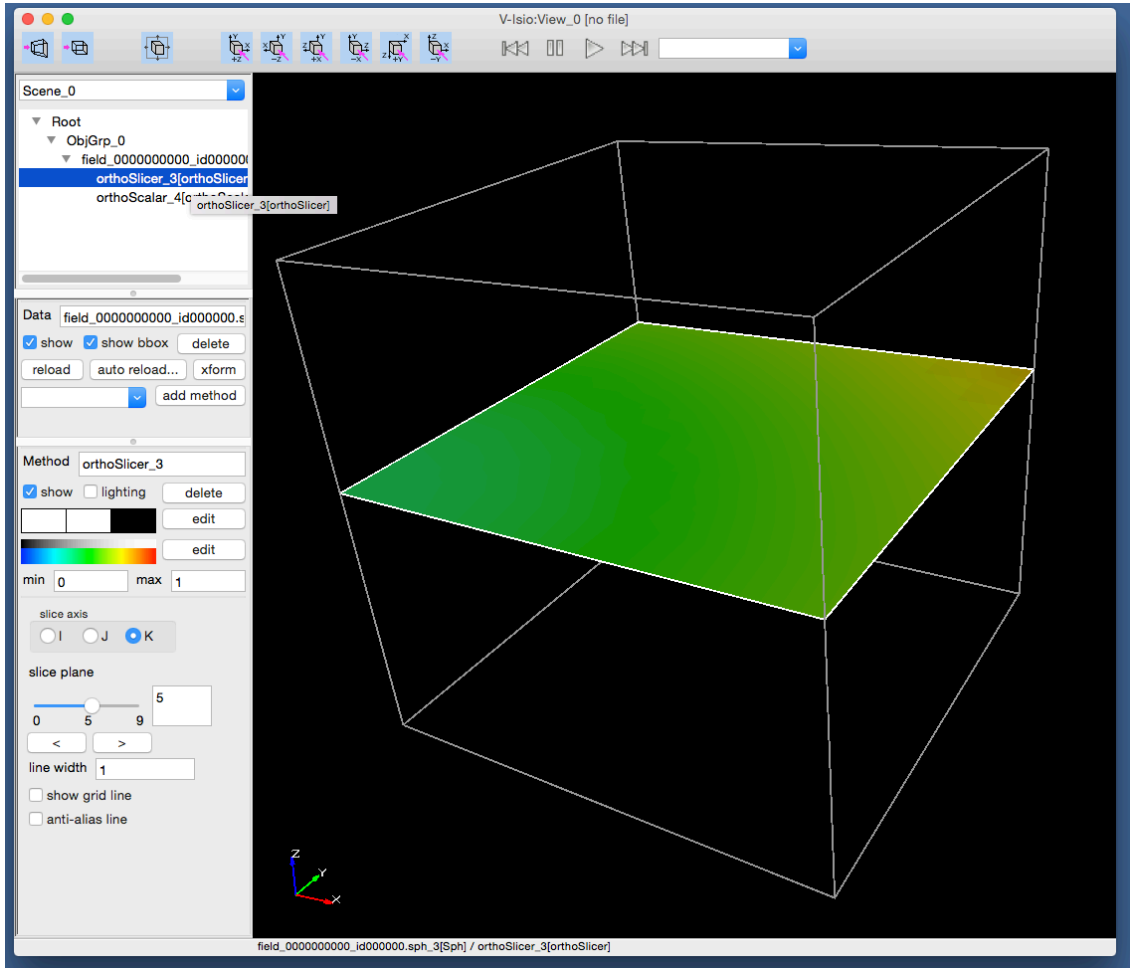


断面制御

オブジェクトツリーで追加されたorthoSlicerを選択すると、メソッドペインに断面制御UIが現れる。スイープする断面方向と断面位置をスライダと< >ボタンで操作する。

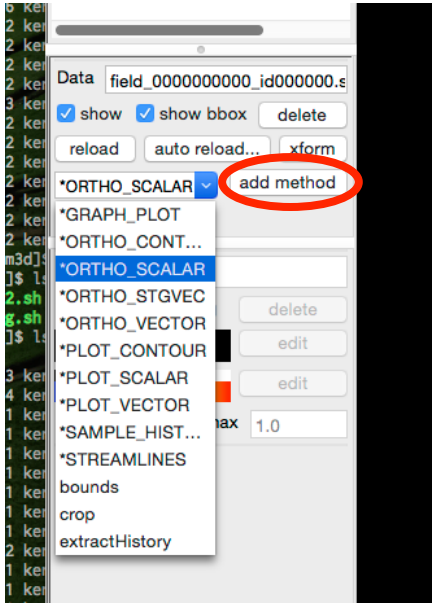
メソッドペインのshowチェックボックスのチェックを外すと、メソッドが無効になる。

メソッドペインのdeleteは、追加したメソッドを削除する。データ操作ペインにもdeleteがあるので注意。



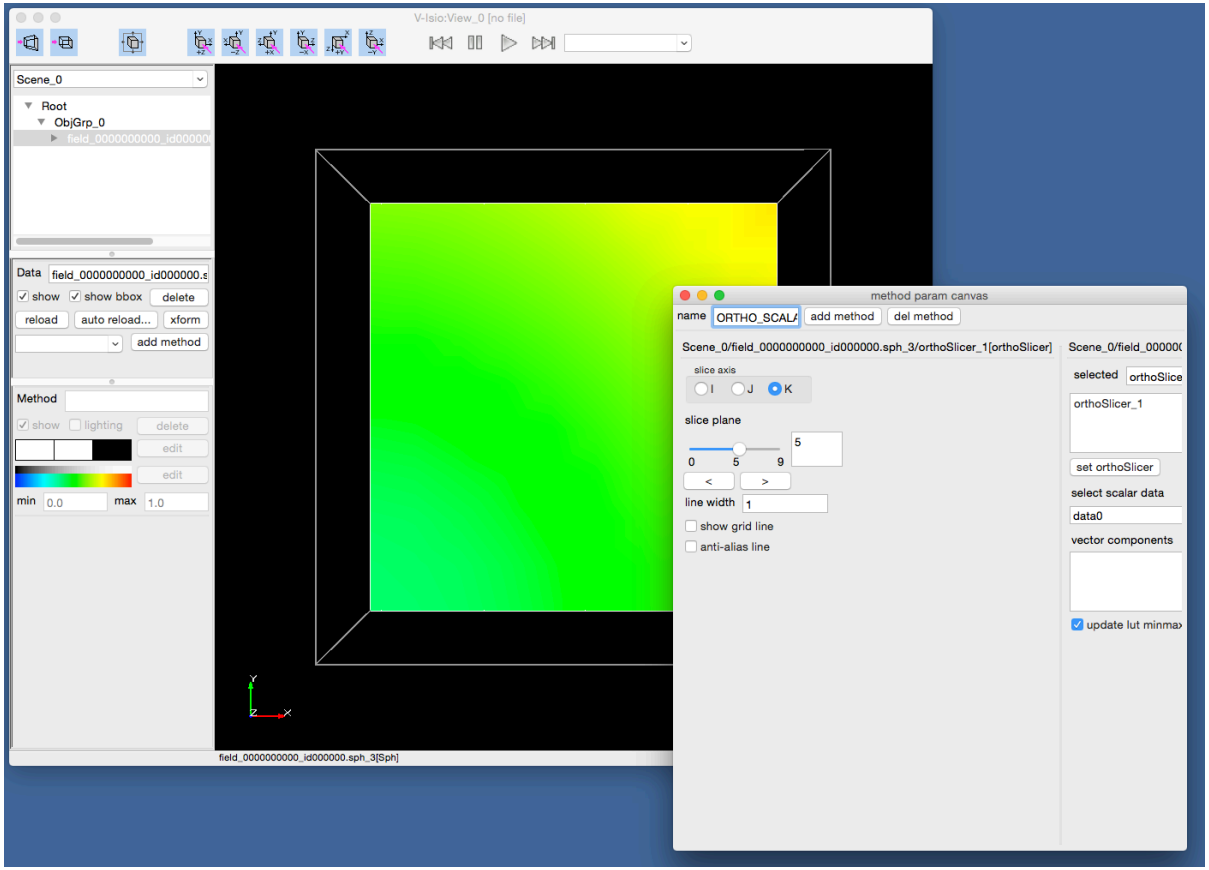
断面可視化の別のセットアップ

データ操作ペインのプルダウンメニューから、インポートしたデータに適用する可視化メソッドを選択する。ここでは、*ORTHO_SCALARを選択。選択後に、add methodをクリックする。



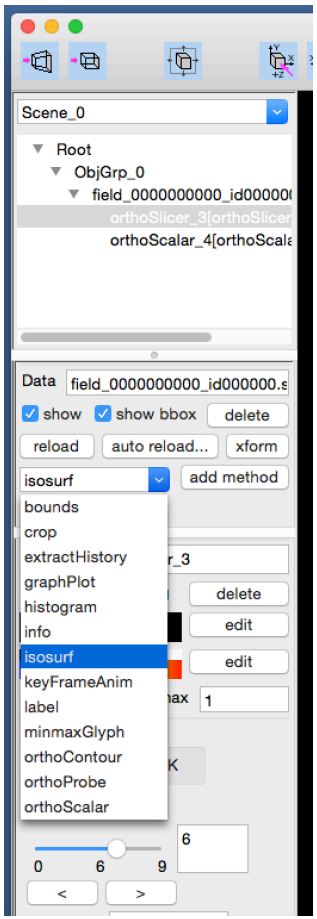
*ORTHO_SCALAR = orthoSlicer + orthoScalar

method param canvasウインドウが現れる。method param canvasは、パラメータ操作のウインドウで、左側でスライス断面を制御する。右側は対象データの選択。メソッドペインでもパラメータ指定は可能。

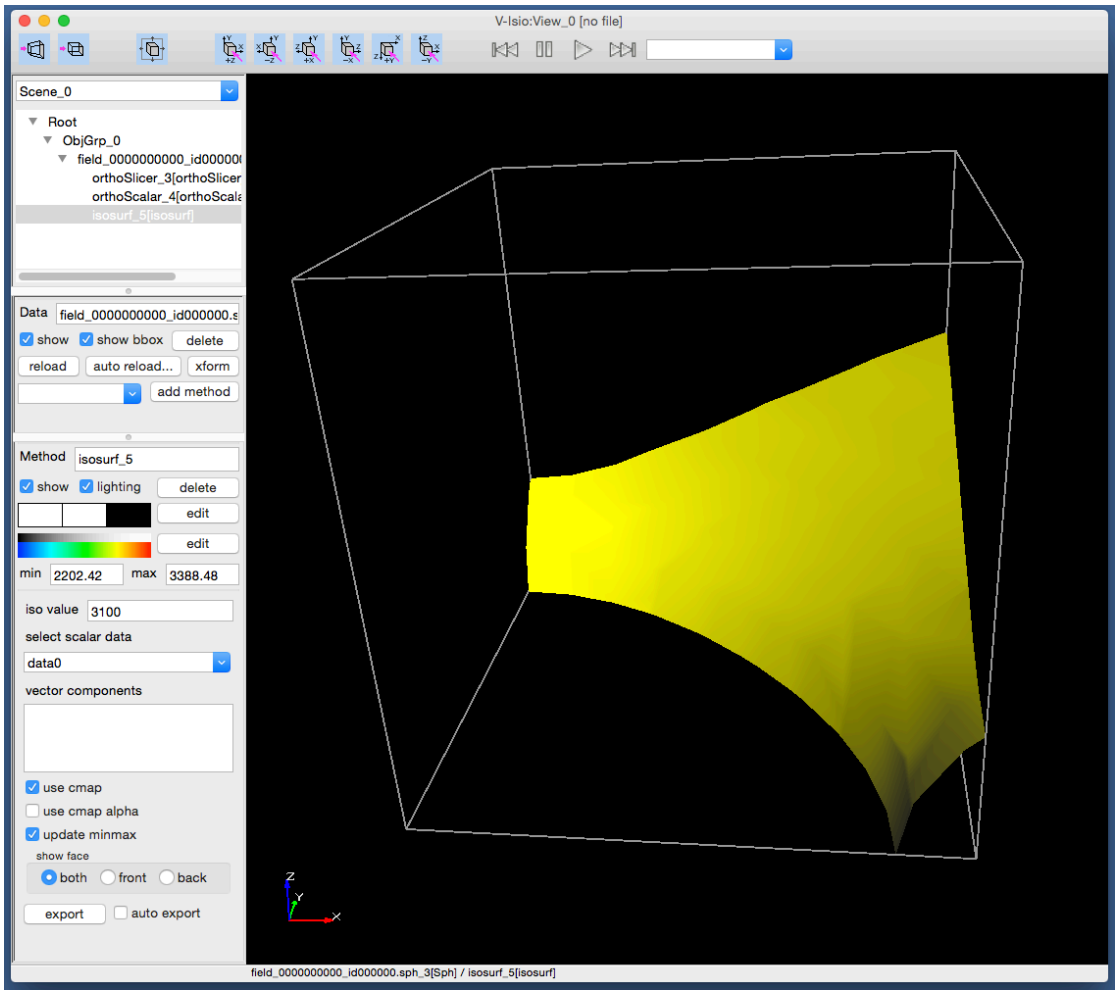


等値面 isosurf

データ操作ペインでisosurfメソッドを追加する。

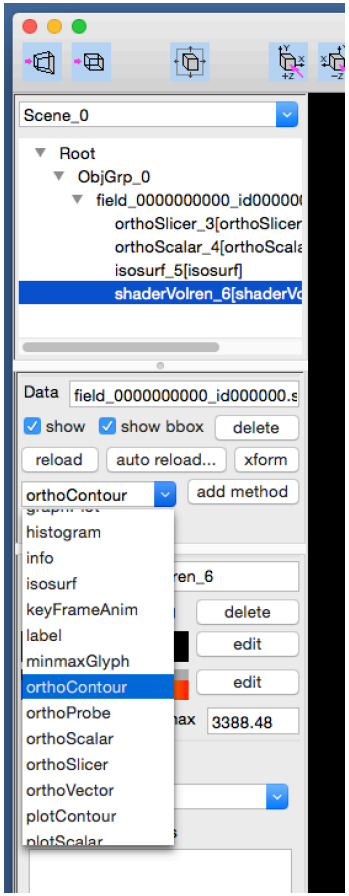


メソッドペインで、iso valueに表示したい等値面の値を入力する。exportで等値面を STLデータとして出力可能。

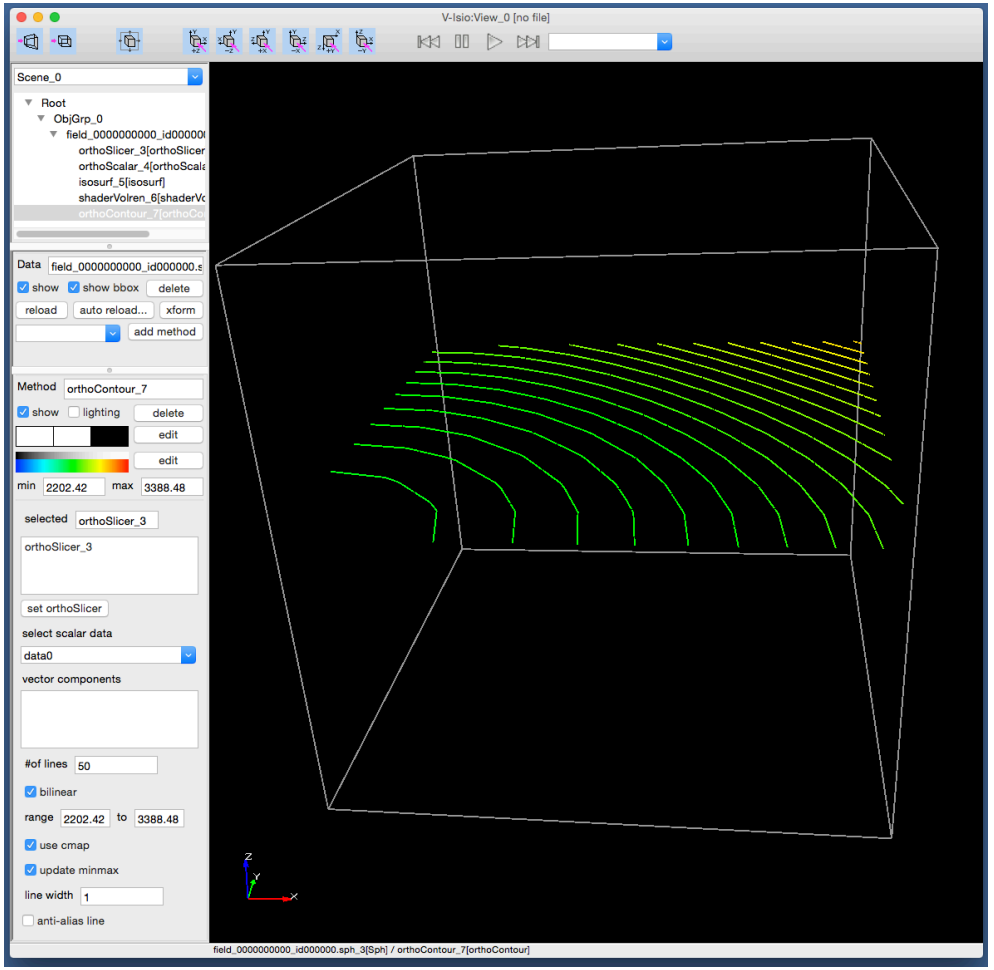


断面の等高線 orthoContour

データ操作ペインでorthoContourメソッドを追加する。

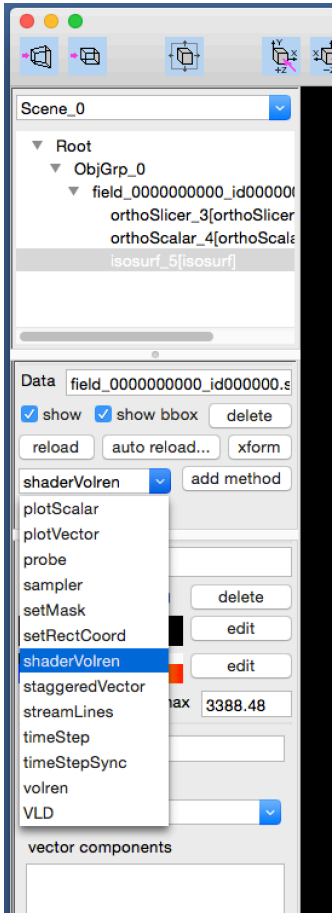


メソッドペインで、スライス断面メソッド、対象データ、等高線の本数などのパラメータを設定する。



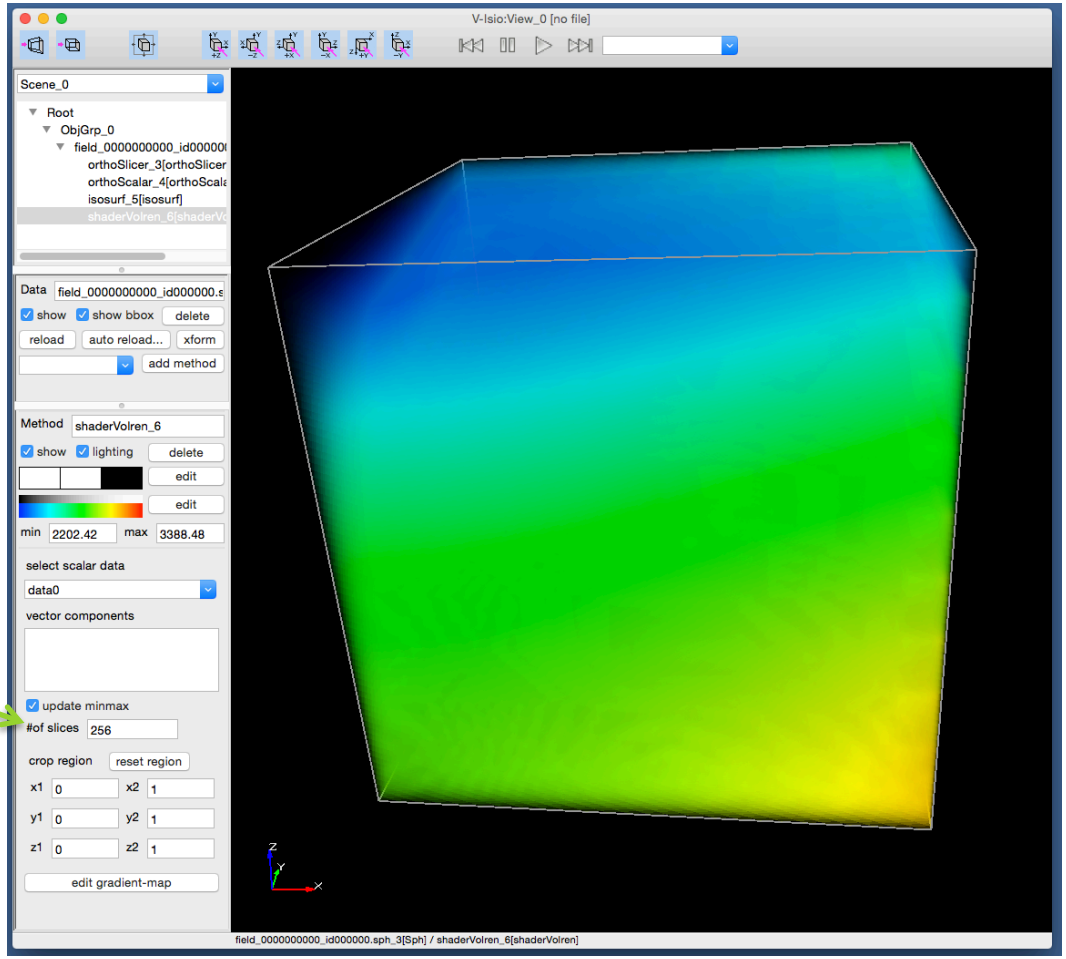
ボリュームレンダリング shadrVolren

データ操作ペインでshaderVolrenメソッドを追加する。

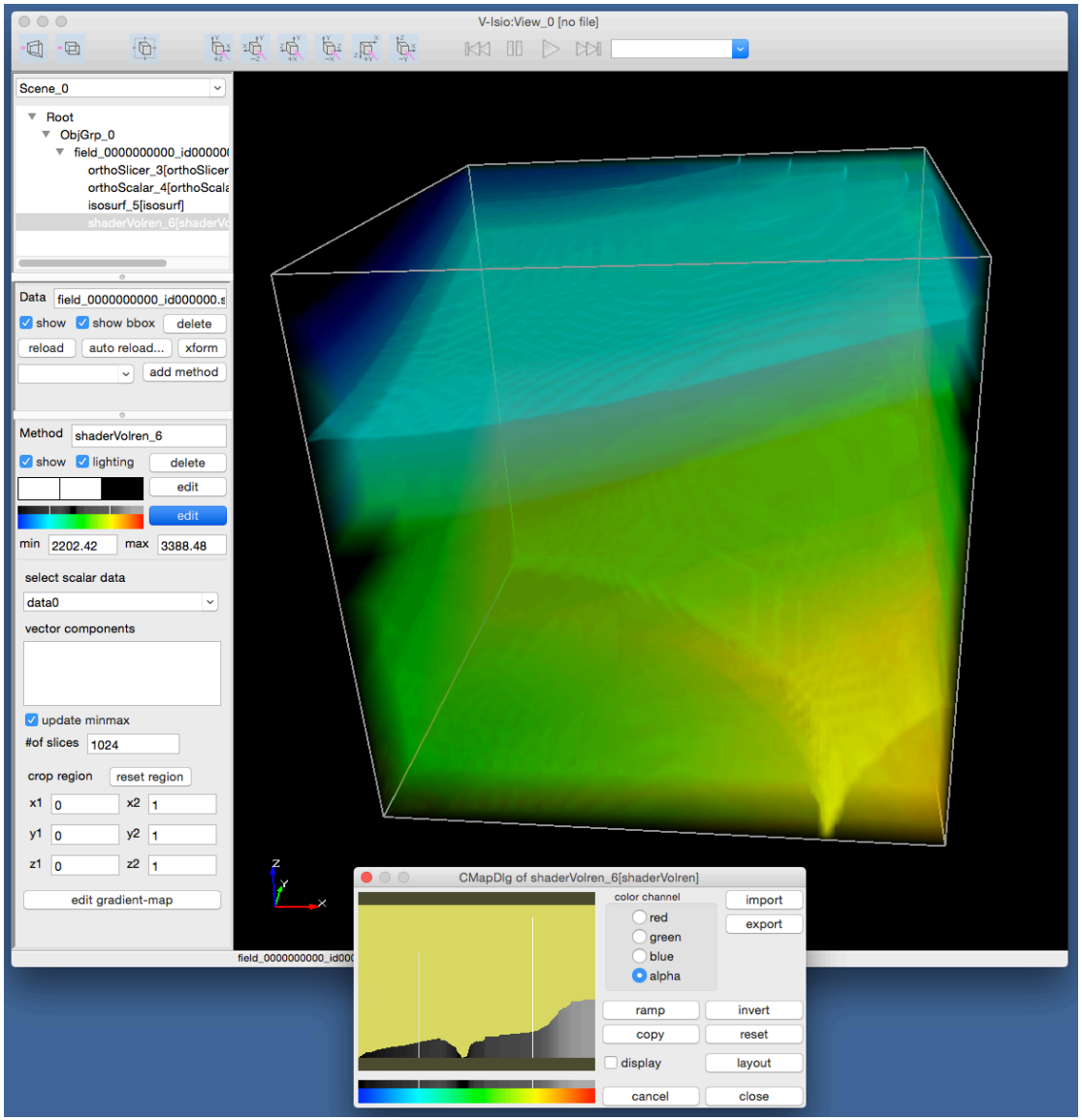


重ね合わせのスライス枚数

メソッドペインで、iso valueに表示したい等値面の値を入力する。exportで等値面を STLデータとして出力可能。

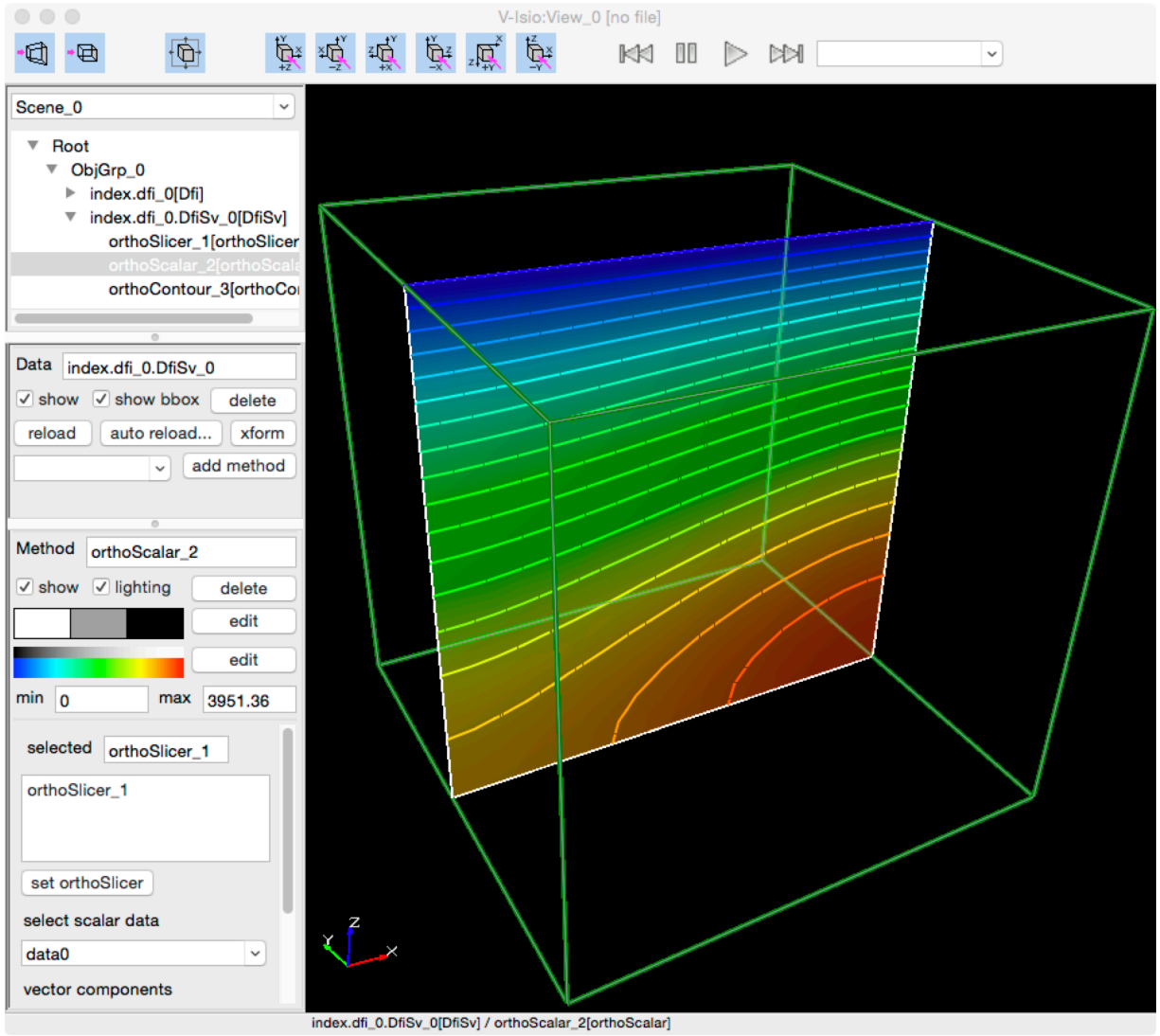


ボリュームレンダリングの伝達関数



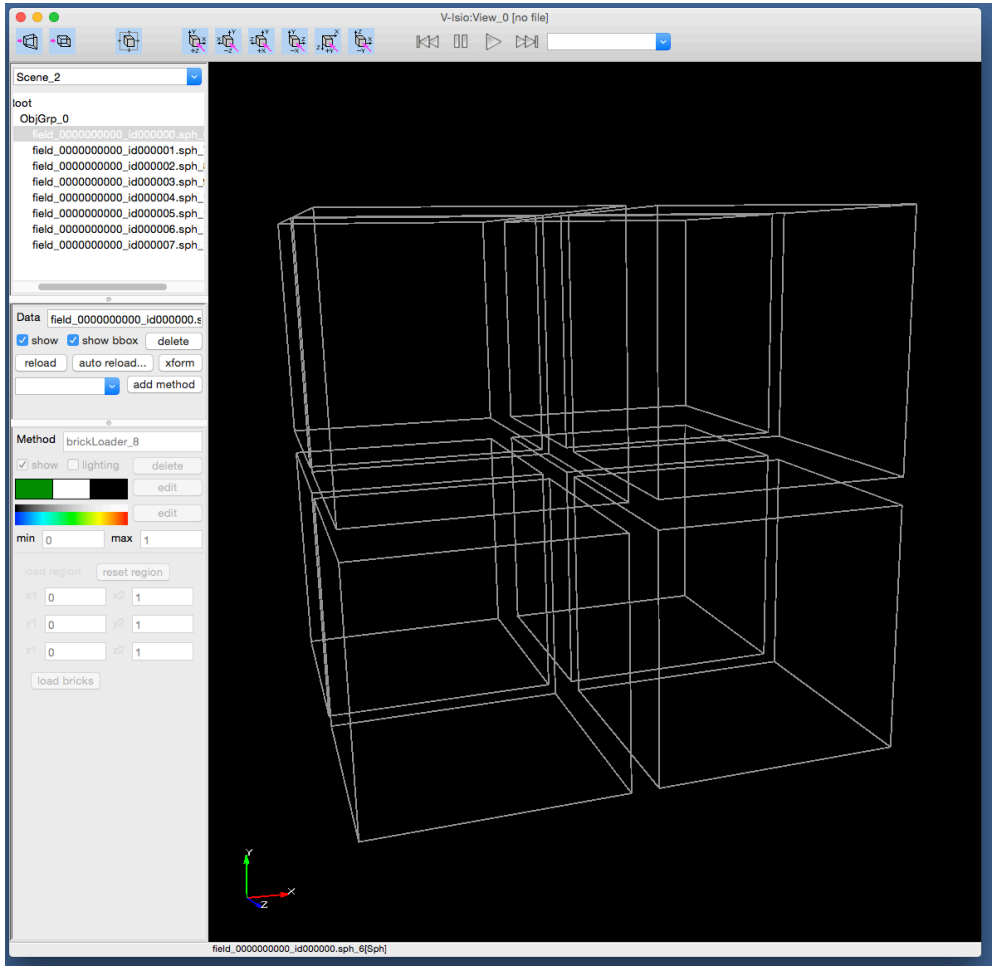
alpha値をマウスでドラッグして設定する。

重ね合わせ



断面の分布と等高線を重ねて表示している。
断面の分布色は、透明度を設定し、半透明にしている。

複数データのロード



File > Import > Sph data files…
で手作業でデータロードは可能

データ数が多くなると大変

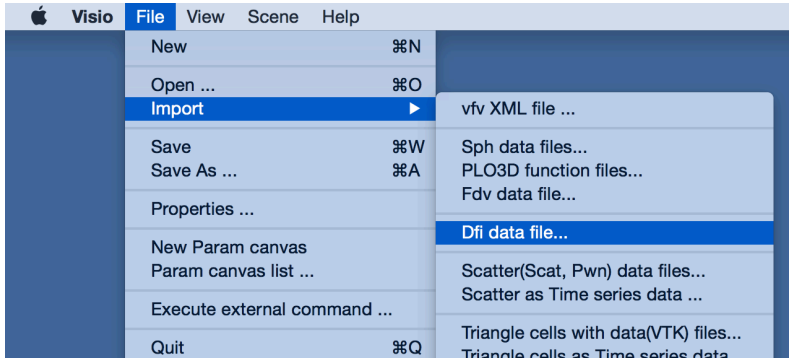
dfiファイルを使った一括ロード

複数データの一括ロード

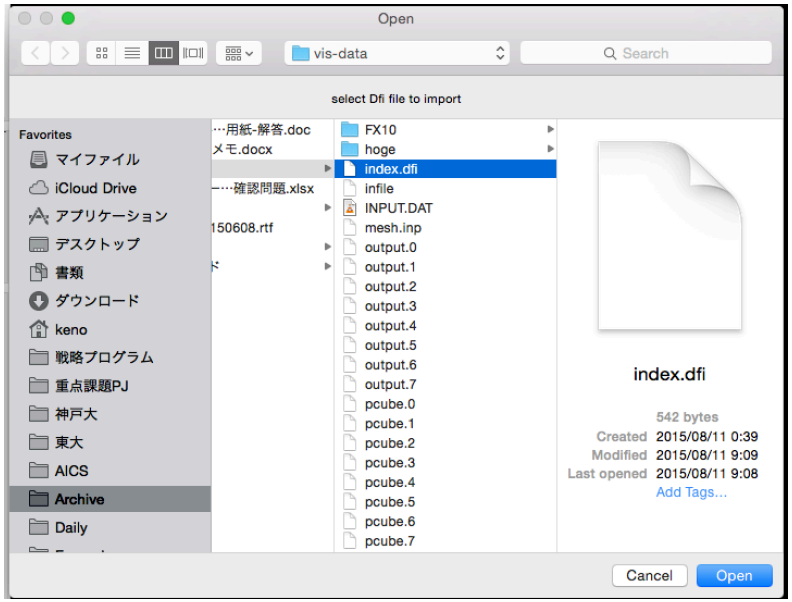
- 並列計算により、各プロセス毎にファイル出力を行うと、複数のファイルが生成される。
- 並列ファイル管理dfiファイルの利用。
  - データの実体＋dfiメタファイル
- V-Isioでdfiファイルを指定して、データの一括ロードを行う。

dfiファイルによるデータロード

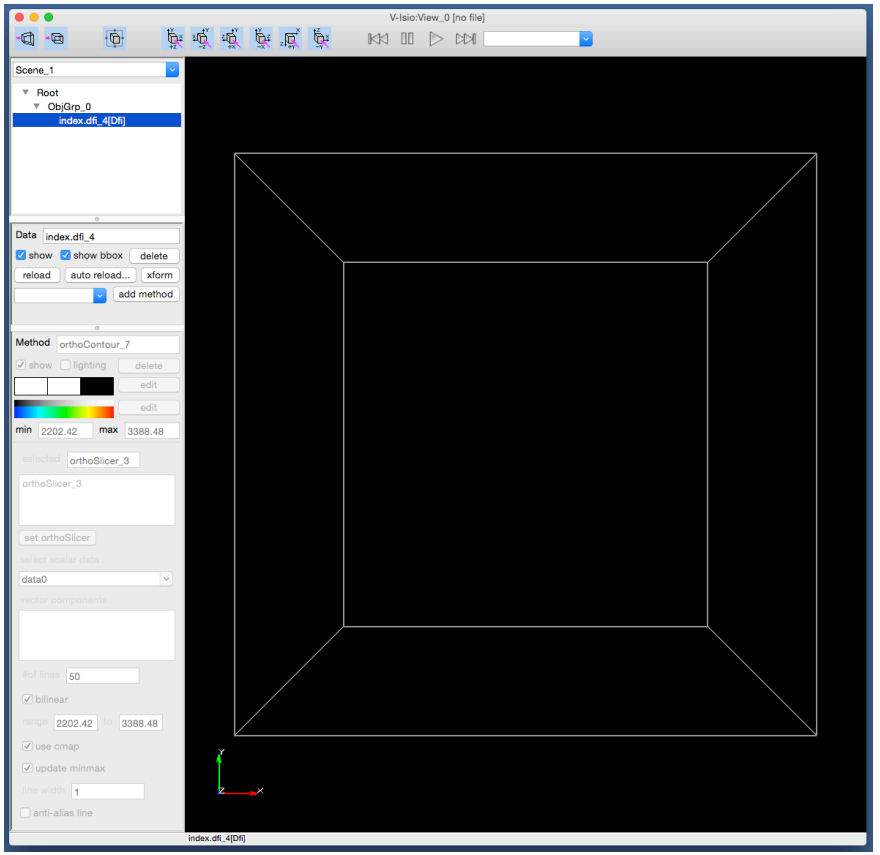
1. FileメニューからDfi data fileを選択



2. index.dfiファイルを選択

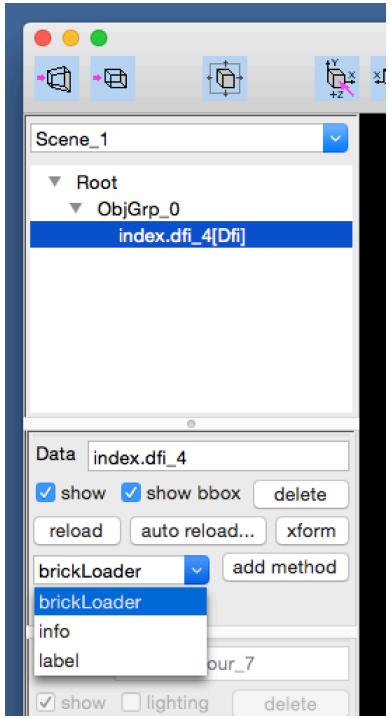


3. index.dfiファイルをロードし、bounding boxなどの情報から領域が表示される。この時点では、データ本体はロードされていない。



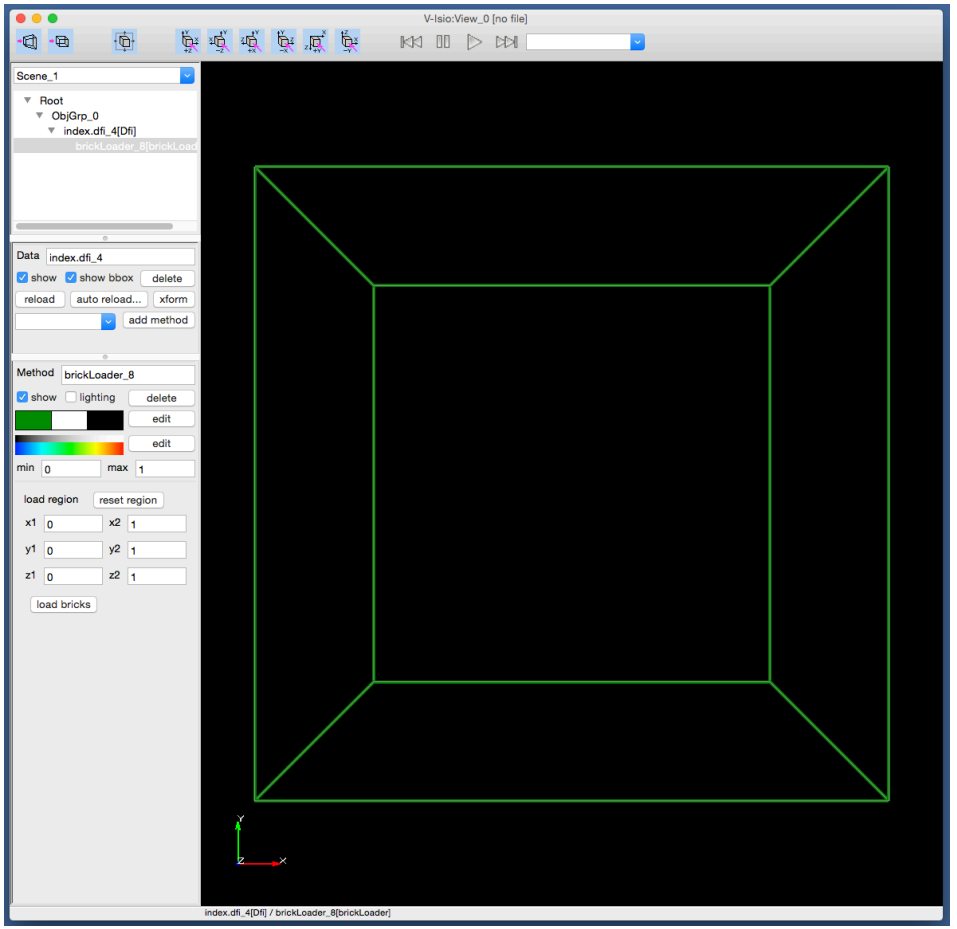
dfiファイルによるデータロード

3. データ操作ペインでbrickloaderを追加する。



この例は、8並列計算で各軸方向に2分割である。
つまり、x={0, 1}, y={0, 1}, z={0, 1}の範囲で選択可能。

4. brickloaderを追加されると、bounding boxの色が緑色に変化する。これは、メソッドペインの領域選択に対応している。load bricksをクリックすると、選択されている領域の実データがロードされる。



dfiファイルによるデータロード

データがロードされると、オブジェクトツリーにDfiSvの文字列を含むデータオブジェクトが現れる。

一旦、データがロードされれば、単一データと同じように可視化が可能になる。