The Usege of Gmsh

2012年11月28日

1　はじめに

Gmsh は，FreeFEM などと同じ，FEM 解析ソフトの一つである．
メッシャーとしての利用を前提としているため，メッシュデータを生成して保存するところまでを目標とする．
２次元メッシュであれば，使い方などの情報収集の点やラベル付けのシンプルさなどの点から，FreeFEM を利用するほうが良いと思われる．しかし３次元メッシュとなると，FreeFEM としての３次元に対する対応力の低さや関連文献が少ない点から，利用をおすすめできない．
３次元であれば，Gmsh を用いればある程度容易に，それなりに複雑な設定のメッシュデータを生成できると思われる．

2　インストール

Gmsh は Linux 環境であれば CUI にてパッケージをインストールするだけで利用できる．Windows，Mac 版もあるので，こちらは次の公式サイトからファイルをダウンロードしてインストール．
公式サイト ： http://geuz.org/gmsh/

3　メッシュ生成

3.1　メッシュ生成手順

Gmsh では，点，線，面，体積などをまとめてエンティティと呼ぶ．点エンティティから初めて，段々と上位のエンティティを生成していく．
また，これらの図形を構成するエンティティを基本エンティティと呼ぶのに対し，基本エンティティに物理的な意味 (ほとんどラベルみたいなもの) を持たせたエンティティを物理エンティティという．
Gmsh は，その操作を GUI で行うことができるが，ここではスクリプトを用いて CUI 上でメッシュを生成していく．つまり，*.geo ファイルを生成して実行するることでメッシュデータを生成する．

メッシュは，分割したい領域を定義し，その領域に対するメッシュ生成コマンドを入力することで生成される．
領域の定義の流れは次のようである．まず最低次エンティティである点 (Point) を，座標を指定して定義し，その点を用いて有向線分 (Line) を定義する．次に，有向線分で閉じられた線ループ (Line Loop) を作り，その線ループ内部を面 (Surface) として定義することで２次元領域を生成する．２次元メッシュであればこれでメッシュ生成コマンドを打てばメッシュ完成であるが，３次元の場合は領域の定義が続く．複数個の面を用いて面ループ (Surface Loop) を作る．これが内部領域を含む閉じた面になっていれば，この内部を体積 (Volume) として定義する．この体積に対してメッシュ生成コマンドを打てば３次元メッシュが生成される．
メッシュ生成の大まかな手順は以上であるが，メッシュサイズの指定や物理エンティティ作成 (ラベル付け)，メッシュデータの読み方などはそのステップで述べていく．
なお，以下の小節同士のエンティティの名前に整合性はないので注意すること．

3.2　変数定義

スクリプトの先頭で，変数を定義しておくと便利である．

```
radius = 4.0;
cellSize=1;
pio2=Pi/2;
```

”Pi”は π として使える．

3.3 基本エンティティ 点 (Point)

点は，名前パラメータを用いて次のように定義する．

```
Point(1) = { 0, 0, 0, cellSize};
Point(2) = {-radius, 0, 0, cellSize};
Point(3) = {0, radius, 0, cellSize};
Point(4) = {0, 0, radius, cellSize};
```

・Point(名前) = {(x 座標), (y 座標), (z 座標)，(要素長基準値) }
要素長基準値はここでは 3.2 で定義した cellsize を用いている．あとで説明するが，この値を小さくしてもメッシュ全体が細かくはならないので，一様なサイズのメッシュを切りたい時はある程度大きめが良い．

3.4 基本エンティティ 線 (Line)

線は，点エンティティを用いて次のように定義する．

```
Line(101) = {1,2} ;
Circle(102) = {2, 1, 3};
Ellipse(113) = {2, 1, 3, 2};
```

・Line(名前) = { (始点)，(終点) }
始点から終点への直線を定義する．

・Circle(名前) = { (始点)，(中心点)，(終点) }
始点 (終点) と中心点間の距離を半径とする，始点から終点までの円弧を定義する．

・Ellipse(名前) = { (始点)，(中心点)，(長軸方向上の点)，(終点) }
始点から，終点へ楕円曲線を定義する．中心点は楕円の中心である．普通，長軸上の点は始点，或いは終点のどちらか長軸側の点にしておけばよい．これは Circle の機能も兼ねている．

3.5 基本エンティティ 線ループ (Line Loop)

3.3 で定義した線分を用いて，線ループを定義する．

```
Line Loop(201) = {101, -103, 102} ;
```

線エンティティは有向線分であることに注意して，左回り，左側が内側になるように線分をつないでループを作る．有向線分の前に”-”をつけると逆向き線分になるので，適宜利用する．

3.6 基本エンティティ 面 (Surface)

面を定義する．

```
Plane Surface(301) = { 201 };
Ruled Surface(302) = { 202 };
```

・Plane Surface(名前) = { (線ループ) }
引数の線ループに対して，その線ループを構成する点エンティティを通るような平面を生成する．

・Ruled Surface(名前) = { (線ループ) }
引数の線ループに対して，その線ループを構成する線分に合わせた曲面を生成する．球面などの曲面の場合はこちらを使う．

3.7 基本エンティティ 面ループ (Surface Loop)

複数の面から構成される内部領域を持った閉じた面を定義する．

```
Surface Loop(401) = {301,302,303,304,305,306};
```

・Surface Loop(名前) = { (要素となる複数の面)，· · · (面)};
空間領域を囲うように面ループを書く．空間が閉じてさえいれば書く順番は関係ない．例えば直方体ならば 6 つの面で面ループを定義することになる．

3.8 基本エンティティ 体積 (Volume)

閉じた面ループの内部を 3 次元領域として定義する．

```
Volume(501) = {401};
Volume(502) = {401,402};
```

・Volume(名前) = { (面ループ) };
面ループの内部全体を領域とするときは単一の面ループで体積を定義する．

・Volume(名前) = { (面ループ)，· · ·，(面ループ) };
ある面ループの内側で，他の面ループの外側というような，3 次元領域の中が一部くり抜かれているような体積を定義したい時は，大外を囲う面ループとくり抜く内部領域を囲う面ループを書く．複数の面ループで囲まれた 3 次元領域が定義される．
繰り抜いた部分は，501 の定義のように単一ループで体積を別に定義できる．
例：Ball in Ball 型，Ball in Cube 型

3.9 物理エンティティ

生成したメッシュデータに，ラベル情報を持たせるためのエンティティである．点・線・面・体積エンティティそれぞれに，ラベルを付けることが出来る．ループには物理エンティティを定義することはできない．1 つでも物理エ

ンティティを定義すると，最終的に生成されるメッシュデータファイルには定義した物理エンティティに関する情報のみが記述される．この点についてはメッシュデータの章で詳しく述べるので，ここでは定義の仕方のみについて述べる．

```
Phisical Point(1001) = { 1,2,3,4 }
Phisical Line(1002) = { 101,102,103 }
Physical Surface(1003) = {301,t1[0],t2[0],t3[0],t4[0],t5[0],t6[0],t7[0]};
Physical Volume(1004) = {501};
```

基本的には，付けたいラベル番号を左辺に，ラベルを付けたいエンティティを引き数に記述するだけである．
物理エンティティのレベルと基本エンティティのレベルは一致しないといけない．(Phisical Point に対しては点エンティティ 1,2,3,4 を引数にするなど)

3.10 エンティティの移動・コピー

3 次元メッシュを生成するときは，内部領域を囲うように必要数の面を生成して面ループを定義しないといけないが，この作業をすべて記述すると煩雑なので，平行移動，回転移動，及びそれらのコピー機能の使い方について述べる．この機能を使うと，例えば球面であれば 1/8 球面を生成して回転移動コピーを 7 つ配置することで球面を生成するのに必要な 8 つの面を準備することができる．また，直方体であれば，1 つの面を生成すれば，それの回転移動コピー及び平行移動コピーを行い，必要な 6 面を生成できる．
平行移動ならば Translate，回転移動ならば Rotate，コピーならば Duplicata を用いる．

```
Translate {1, 0, 0} { Surface{301} }
t1[] = Translate {1, 0, 0} {Duplicata{Surface{301};}};
t2[] = Rotate {{0,0,1},{0,0,0},pio2} {Duplicata{Surface{301};}};
```

1 行目は，面 301 を x 軸正の方向に 1 だけ平行移動している．この場合，エンティティの名前はそのまま引き継がれる．
t1[] は，面 301 のコピーを，x 軸正の方向に 1 だけ移動しだ場所に配置している．
t2[] は，面 301 のコピーを，{0，0，0} を回転の中心，ベクトル {0，0，1} を (今は z 軸) 回転軸として，$\pi/2$ だけ移動しだ場所に配置している．pio2 は 3.2 での定義に従う変数.
移動及びコピーにより生成されるエンティティには移動とは異なり名前を付ける必要があり，名前は”t1[]”のように，配列の形で定義されることに注意する．また，コピーにより生成したエンティティを用いて次のエンティティを定義するときは”t1[0]”のように，配列の 0 番目要素を用いる．
Surface Loop(401) = {301,t1[0],t2[0]};
のように書くことで，自分で定義した 301 と同じように扱うことができる．

3.11 スクリプト例

球領域を生成するスクリプト例 (sphere.geo) である．

```
//sphere.geo

radius = 5.0;
cellSize=0.3;
pio2=Pi/2;

Point(1) = {0, 0, 0, cellSize};
Point(2) = {-radius, 0, 0, cellSize};
Point(3) = {0, radius, 0, cellSize};
Point(4) = {0, 0, radius, cellSize};

Circle(101) = {2, 1, 3};
Circle(102)= {4,1,2};
Circle(103)= {4,1,3};

Line Loop(201) = {101, -103, 102};

Ruled Surface (301) = {201};

t1[] = Rotate {{0,0,1},{0,0,0},pio2} {Duplicata{Surface{301};}};
t2[] = Rotate {{0,0,1},{0,0,0},pio2*2} {Duplicata{Surface{301};}};
t3[] = Rotate {{0,0,1},{0,0,0},pio2*3} {Duplicata{Surface{301};}};
t4[] = Rotate {{0,1,0},{0,0,0},-pio2} {Duplicata{Surface{301};}};
t5[] = Rotate {{0,0,1},{0,0,0},pio2} {Duplicata{Surface{t4[0]};}};
t6[] = Rotate {{0,0,1},{0,0,0},pio2*2} {Duplicata{Surface{t4[0]};}};
t7[] = Rotate {{0,0,1},{0,0,0},pio2*3} {Duplicata{Surface{t4[0]};}};

Surface Loop(401)={301,t1[0],t2[0],t3[0],t4[0],t5[0],t6[0],t7[0]};

Volume(501)={401};

Physical Surface(1001) = {301,t1[0],t2[0],t3[0],t4[0],t5[0],t6[0],t7[0]};
Physical Volume(1002) = {501};
```

※スクリプト内においては，C ソースと同様のコメントアウトが利用できる．

ここでは半径 (radius) 5，要素長基準値 (cellSize)0.3 として，球を生成している．Point(·) にて球の中心 1 と 1/8 球面上の 3 点 2，3，4 の合わせて 4 点を定義している．
線エンティティ 101〜103 は 1/8 球面を囲う枠の要素となる線分である．
線ループ 201 で，1/8 球面を囲う枠を生成している．
面 301 は，201 で閉じている，球面に沿った面である．
ここまでで，図 1 のようなジオメトリが生成されている．301 は面として定義されているので，図 2 のように 1/8 球

面メッシュを生成することもできる．

次に，全球面を生成するための残りの 1/8 球面 7 つ，t1[]〜t7[] を回転移動にて定義している．
以上の 8 つの 1/8 球面を用いて，全球面ループ 401 を生成している．
以上の作業の結果が図 3 である．ここまでで，図 4 のような球面メッシュの生成が可能である．
最後に体積エンティティ 501 を定義することで，球の内部領域に対する 3 次元メッシュ分割が行える．
最後に，物理エンティティを定義している．ここでは球面をなす三角形要素にラベル 1001 を，内部領域をなす四面体要素にラベル 1002 をつけている．これらはメッシュデータに反映される．

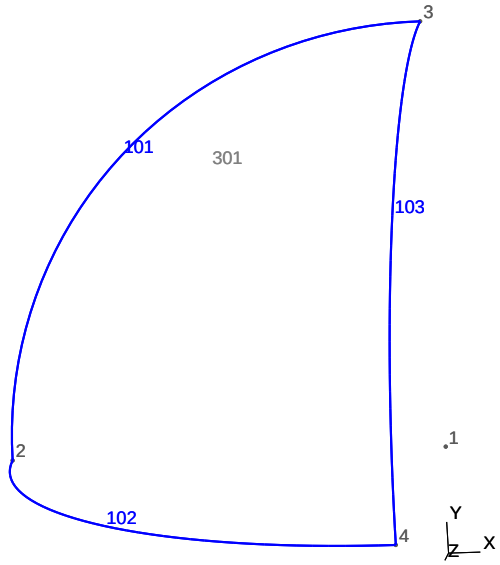

図 1

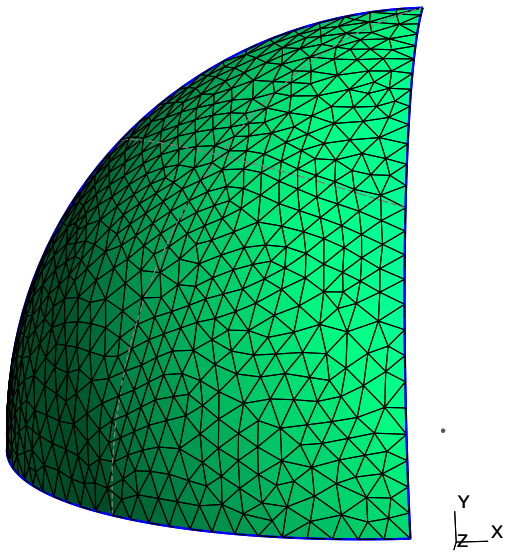

図 2

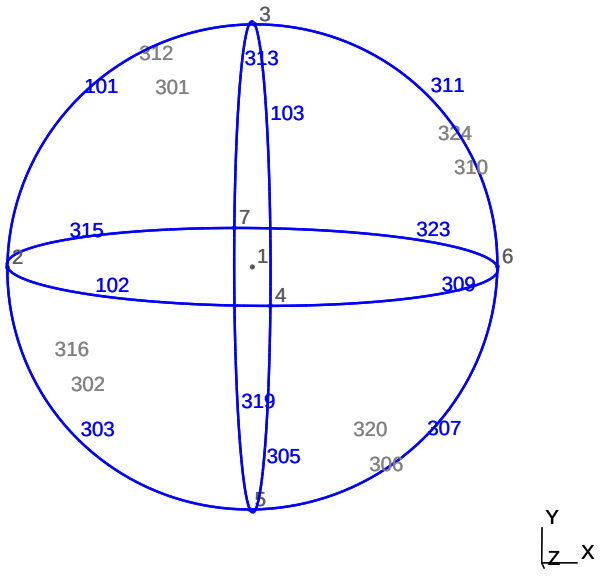

図 3

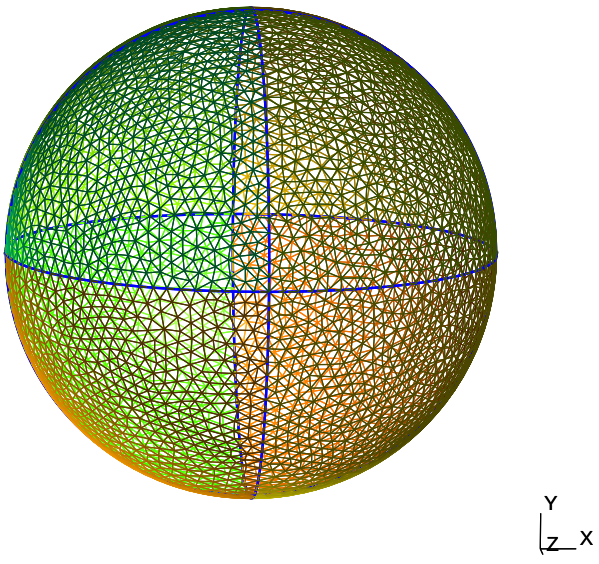

図 4

3.12 メッシュ分割

3.10 のスクリプト sphere.geo を例にとって説明する．
sphere.geo を GUI で呼び出すには，コマンドラインで
”gmsh sphere.geo”
とすれば，グラフィックウィンドウに図 3 が描かれる．”Mesh”→”3D(2D)”とすれば，メッシュ分割が行われる．2D とすれば表面メッシュ，3D とすれば内部領域に対する 3 次元メッシュを生成する．更に”Mesh”→”Save” で，メッシュデータの保存ができる．

コマンドラインでのメッシュ分割は，
”gmsh sphere.geo -3”
とすれば，引数にしたファイルと同じ名前のメッシュデータファイルが生成される．オプションの”-3”は，3 次元を表し，”-2”とすると表面メッシュが生成する．ここでは sphere.msh というファイルが生成される．
図 5 は生成した 3 次元メッシュの図である．

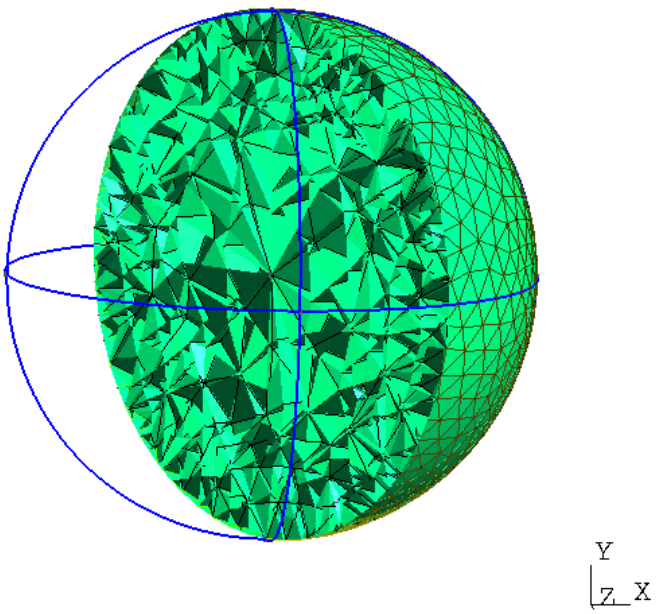

図 5

若干見にくいが，内部までメッシュ分割されていることが確認できる．ここで注意すべきは，Gmsh は不均一メッシュを生成する点である．点エンティティを定義するときに，要素長基準値を設定したが，この値が反映されるのはその点から生成される面エンティティまでであり，球の中心付近 (近傍にエンティティが定義されてない部分) はメッシュが荒いことが分かる．

領域全体で均一なメッシュを生成したい時は，既存のメッシュに対するリファイン (リ・メッシュ) 機能を使うのが良い．リファイン機能は既存のメッシュに対して要素長が 1/2 程度のメッシュを再生成する．
球などの場合は球面に沿ってリメッシュするので，曲面はきちんと真球に近づいていく．

この場合は要素長基準値を 2 などとすれば，最初のメッシュ分割がある程度一様になるので，それを用いて逐次的

に細かいメッシュを再生成する．
GUI では，”Mesh”→”refine”とすると，既存のメッシュを再分割してくれる．
コマンドラインでは
”gmsh -refine sphere.msh”
とすると，より細密なメッシュに置き換わる．ここではメッシュデータファイルを引数としていることに注意する．

基本的に，荒いメッシュを生成しておいて，必要な細かさのメッシュになるまでリファインを繰り返す

4 メッシュデータファイル

メッシュ分割に Gmsh を用いると，”*.msh”というメッシュデータファイルが生成される．まずはメッシュデータファイルの例を示す．下は sphere.geo から生成したメッシュデータである．

```
$ MeshFormat
2.2 0 8
$ EndMeshFormat
$ Nodes
39
1 -5 0 0
2 0 5 0
3 0 0 5
...
39 -1.46711972495516 -1.486957954076984 -1.480137539178151
$ EndNodes
$ Elements
172
1 2 2 1001 301 10 1 7
2 2 2 1001 301 8 10 7
...
56 2 2 1001 324 24 23 30
57 4 2 1002 501 32 5 31 34
58 4 2 1002 501 6 32 31 34
...
172 4 2 1002 501 12 21 22 33
$ EndElements
```

”$Mesh Format”については，特に利用しないので触れない．
”$Nodes”では，一行目に総節点数があり，ここでは 39 である．次の行から各節点の節点番号，x 座標，y 座標，z 座標が記されている．

”$Elements”では，定義した物理エンティティについてのみ記述されている．つまり，要素節点対応表が必要ならば，体積の物理エンティティを定義する必要がある．Dirichlet 境界条件を課す場合は，その境界面についても物理エンティティを定義する必要がある．
・(要素番号) (要素タイプ) (ラベル数) (ラベル) (ラベル) (節点リスト)
の形で与えられる．
要素タイプは，主に利用するもののみ書いておく．
15：点要素，1：線要素，2：三角形要素，3：2 次元四角形要素，4：四面体要素

要素タイプの一覧については Gmsh の公式マニュアルにある．
ラベル数は，たいていの場合 2 であり，2 つのラベルを付けることが出来るが，ここではラベルは 1 つあれば十分なので，2 つ目以降のラベルについては述べない．1 つ目のラベルに物理エンティティの定義で用いた 1001 と 1002 という番号が振られているのが分かる．
節点リストは，三角形要素ならば 3 点，四面体要素ならば 4 点の節点番号が並んでいる．
Dirichlet 境界条件を課す場合は，節点要素 (要素タイプ 15) でラベルを読みたいが，面上の点に対して物理エンティティを定義できないので，面上の三角形要素にラベルをつけて，メッシュデータを読み込むプログラムの中で適宜その三角形要素を構成する節点のラベルに読み替える必要がある．

要素番号は，定義した物理エンティティ全て含めて通し番号で与えられているので，要素数として四面体要素数が欲しい場合は要素番号 57〜172 の計 126 が四面体要素数となる．この値についても，自分のプログラムの中で計算しないといけない．

物理エンティティを定義することを前提としているのは，定義しなかった場合に”Elements”に点・線分・三角形・四面体要素のすべてが記述されるので，必要なデータの読込が面倒になる上にファイルサイズも大きくなるからである．

5 サンプル

最後に，ここに記述したことを組み合わせて作成した，球領域内に楕円体領域を含むメッシュデータを生成するためのスクリプトを載せる．

```
/*
0** 点 (Piont) 要素
1** 線 (Line,Crrcle) 要素
2** 線ループ (Line Loop)
3** 面 (Surface) 要素
4** 面ループ (Surface Loop)
5** 体積要素 (Volume)
*/

radius = 4.0;
radius2 = 2.0;
cellSize=1;
cellSize2=1;
pio2=Pi/2;

// /*
// 外球用　点・線定義
// */

// create oval 1/8 shell
Point(1) = {0, 0, 0, cellSize};
Point(2) = {-radius, 0, 0, cellSize};
Point(3) = {0, radius, 0, cellSize};
Point(4) = {0, 0, radius, cellSize};
Circle(101) = {2, 1, 3};
Circle(102)= {4,1,2};
Circle(103)= {4,1,3};
Line Loop(201) = {101, -103, 102} ; Ruled Surface(301) = {201};

// /*
// 内部領域用　点・線定義
// */

// create inner 1/8 shell
```

```
Point(11) = {-radius2/4, 0, 0, cellSize};
Point(12) = {0, radius2, 0, cellSize};
Point(13) = {0, 0, radius2, cellSize};
Point(14) = {0, -radius2, 0, cellSize};
Point(15) = {0, 0, -radius2, cellSize};

Ellipse(111) = {11, 1, 12, 12};
Ellipse(112) = {13, 1, 13, 11};
Ellipse(113) = {12, 1, 13, 13};
Ellipse(114) = {14, 1, 14, 11};
Ellipse(115) = {13, 1, 13, 14};
Circle(116) = {14,1,15};
Circle(117) = {15,1,12};
Ellipse(118) = {11,1,15,15};

//楕円の Line Loop
Line Loop(211) = {111, 113, 112} ;
Line Loop(212) = {115, 114, -112} ;
Line Loop(213) = {118, 117, -111} ;
Line Loop(214) = {116, -118, -114} ;

//楕円の Surface
Ruled Surface(311) = {211};
Ruled Surface(312) = {212};
Ruled Surface(313) = {213};
Ruled Surface(314) = {214};

// /*
// 各領域の表面ループ定義
// */

// 1/8 shell を元にして，回転移動による全球面 (302〜308) の生成
a302[] = Rotate {{0,0,1},{0,0,0},pio2} {Duplicata{Surface{301};}};
a303[] = Rotate {{0,0,1},{0,0,0},pio2*2} {Duplicata{Surface{301};}};
a304[] = Rotate {{0,0,1},{0,0,0},pio2*3} {Duplicata{Surface{301};}};
a305[] = Rotate {{0,1,0},{0,0,0},-pio2} {Duplicata{Surface{301};}};
a306[] = Rotate {{0,0,1},{0,0,0},pio2} {Duplicata{Surface{a305[0]};}};
a307[] = Rotate {{0,0,1},{0,0,0},pio2*2} {Duplicata{Surface{a305[0]};}};
a308[] = Rotate {{0,0,1},{0,0,0},pio2*3} {Duplicata{Surface{a305[0]};}};

//楕円 1 の全球面生成
b315[] = Rotate {{0,0,1},{0,0,0},pio2*2} {Duplicata{Surface{311};}};
b316[] = Rotate {{0,0,1},{0,0,0},pio2*2} {Duplicata{Surface{312};}};
b317[] = Rotate {{0,0,1},{0,0,0},pio2*2} {Duplicata{Surface{313};}};
b318[] = Rotate {{0,0,1},{0,0,0},pio2*2} {Duplicata{Surface{314};}};

//楕円 2 の全球面生成
b321[] = Translate {2, 0, 0} { Duplicata{ Surface{311}; } };
b322[] = Translate {2, 0, 0} { Duplicata{ Surface{312}; } };
b323[] = Translate {2, 0, 0} { Duplicata{ Surface{313}; } };
b324[] = Translate {2, 0, 0} { Duplicata{ Surface{314}; } };
b325[] = Translate {2, 0, 0} { Duplicata{ Surface{b315[0]}; } };
b326[] = Translate {2, 0, 0} { Duplicata{ Surface{b316[0]}; } };
b327[] = Translate {2, 0, 0} { Duplicata{ Surface{b317[0]}; } };
b328[] = Translate {2, 0, 0} { Duplicata{ Surface{b318[0]}; } };

//楕円 3 の全球面生成
b331[] = Translate {-2, 0, 0} { Duplicata{ Surface{311}; } };
b332[] = Translate {-2, 0, 0} { Duplicata{ Surface{312}; } };
b333[] = Translate {-2, 0, 0} { Duplicata{ Surface{313}; } };
b334[] = Translate {-2, 0, 0} { Duplicata{ Surface{314}; } };
b335[] = Translate {-2, 0, 0} { Duplicata{ Surface{b315[0]}; } };
b336[] = Translate {-2, 0, 0} { Duplicata{ Surface{b316[0]}; } };
b337[] = Translate {-2, 0, 0} { Duplicata{ Surface{b317[0]}; } };
b338[] = Translate {-2, 0, 0} { Duplicata{ Surface{b318[0]}; } };

//Surface Loop の生成
```

```
Surface Loop(401) = {301,a302[0],a303[0],a304[0],a305[0],a306[0],a307[0],a308[0]};//外周
Surface Loop(411) = {311,312,313,314,b315[0],b316[0],b317[0],b318[0]};//楕円 1
Surface Loop(412) = {b321[0],b322[0],b323[0],b324[0],b325[0],b326[0],b327[0],b328[0]};//楕円 2
Surface Loop(413) = {b331[0],b332[0],b333[0],b334[0],b335[0],b336[0],b337[0],b338[0]};//楕円 3

// /*
// 各立体の定義
// */
Volume(501) = {401,411,412,413};
Volume(511) = {411};
Volume(512) = {412};
Volume(513) = {413};


// /*
// ラベル付け
// */
Physical Surface(1001) = {301,t302[0],t303[0],t304[0],t305[0],t306[0],t307[0],t308[0]};
Physical Surface(1002) = {311,312,313,314,b315[0],b316[0],b317[0],b318[0]};
Physical Surface(1003) = {b321[0],b322[0],b323[0],b324[0],b325[0],b326[0],b327[0],b328[0]};
Physical Surface(1004) = {b331[0],b332[0],b333[0],b334[0],b335[0],b336[0],b337[0],b338[0]};
Physical Volume(2001) = {501};
Physical Volume(2002) = {511};
Physical Volume(2003) = {512};
Physical Volume(2004) = {513};
```

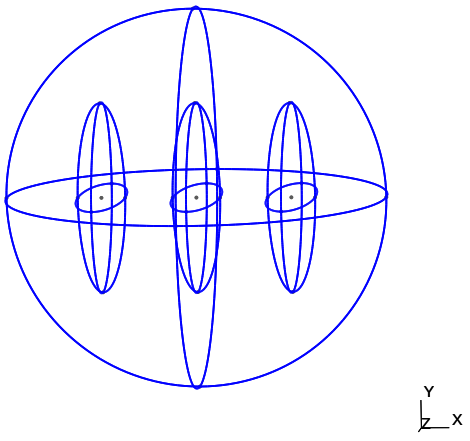

図 6

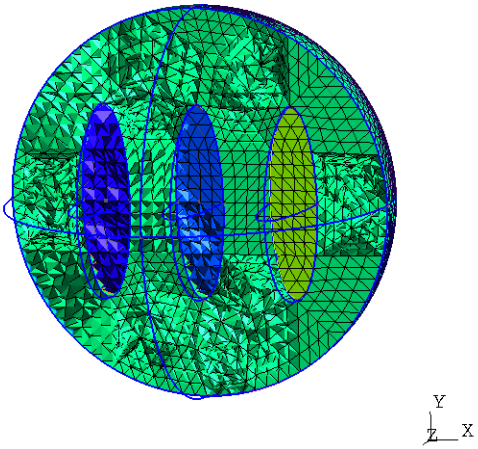

図 7

領域は 1 つの球 $C1$ と 3 つの楕円体 $C2$，$C3$，$C4$ からなる領域である．

$$
\begin{aligned}
C1 \quad &: \quad x^2 + y^2 + z^2 &&= \quad 25 \\
C2 \quad &: \quad 4x^2 + \frac{y^2}{4} + \frac{z^2}{4} &&= \quad 1 \\
C3 \quad &: \quad 4(x-2)^2 + \frac{y^2}{4} + \frac{z^2}{4} &&= \quad 1 \\
C4 \quad &: \quad 4(x+2)^2 + \frac{y^2}{4} + \frac{z^2}{4} &&= \quad 1
\end{aligned}
$$

ラベルは，$C1$ の表面の三角形要素に 1001，C 2，$C3$，$C4$ の表面の三角形要素がそれぞれ 1002，1003，1004，$C1$ の内部で，楕円体の外側にある四面体要素が 2001，C 2，$C3$，$C4$ の内部の四面体要素がそれぞれ 2002，2003，2004 となっている．

6 参考

1	Gmsh 公式サイト	http://geuz.org/gmsh/
2	チュートリアルムービー	http://geuz.org/gmsh/screencasts/
3	Gmsh クイックスタートガイド	http://www.openacoustics.org/wp-content/uploads/2010/09 /gmshQuickstart20100919.pdf

1··· 公式マニュアル (英語) などあり．細かい仕様はこちらで確認できる．

2··· GUI で，メッシュと切るところまでを見られる．とてもわかりやすい・

3··· 公式マニュアルのチュートリアルの一部を日本語に翻訳してある．コマンドに逐一説明がついているのでよい．