文部科学省次世代ＩＴ基盤構築のための研究開発
「イノベーション基盤シミュレーションソフトウェアの研究開発」

CISS フリーソフトウェア

マルチ力学シミュレータ REVOCAP

モデル細分化ツール

REVOCAP_PrePost Ver. 1.1

ユーザマニュアル

お問い合わせ先
（契約窓口）　（財）生産技術研究奨励会
〒153-8505 東京都目黒区駒場４－６－１
（ソフトウェア管理元）　東京大学生産技術研究所 革新的シミュレーション研究センター
〒153-8505 東京都目黒区駒場４－６－１
Fax：03-5452-6662
E-mail：software@ciss.iis.u-tokyo.ac.jp

目次

1 REVOCAP_Refiner 概要

REVOCAP_Refiner は、マルチ力学シミュレーションソフトウェアの一部として、マルチ力学シミュレーションソフトウェアで開発されるソルバ、カップラに組み込まれて使われ、オンメモリ上で有限要素法、有限体積法の非構造格子の要素を細分するライブラリである。メッシュ生成、境界条件設定、領域分割等の通常のプレ処理が行われた解析モデルについて、ソルバまたはカップラはオンメモリで解析モデルを細分することができる。

単純に細分するだけでは、曲面形状の幾何的な解像度は細分前のメッシュの解像度を越えることができない。REVOCAP_Refiner では、この点にも留意し要素細分のときに、メッシュ生成に用いた CAD モデルを再度用いて、要素細分と同時に形状の補正を行い、細分された解析モデルの CAD モデルに対する形状の解像度を改善する。

2 動作環境

REVOCAP_Refiner の動作環境は、組み込まれて利用されるソルバおよびカップラの動作環境に準ずる。超並列計算機および次世代スパコンの上での動作を想定している。各種スパコン、PC クラスタ、次世代スパコン上 Unix 互換の OS のもとで動作する。動作が確認されている環境は以下の通り。

- SGI Altix (Linux)
- FUJITSU PRIMERGY (Linux)
- HA8000（Linux）
- PC-Cluster (Linux CentOS)
- Earth Simulator2

3 開発言語、コンパイラに関する情報

REVOCAP_Refiner は REVOCAP_PrePost とメッシュ処理部を共通化している。その部分の開発は C++ 言語でなされている。ソルバとのインターフェイスを取る部分は C 言語で開発されている。言語標準の以外の外部ライブラリについては特に利用していない。
動作が確認されているコンパイラは以下の通り。

- g++
- Intel C
- PGI C
- Fujitsu FCC

また、REVOCAP_Refiner を呼び出すソルバ側の Fortran 言語について、動作が確認されているコンパイラは以下の通り。

- gfortran
- Intel Fortan
- PGI Fortran
- Fujitsu Frt

日立コンパイラは namespace の対応でエラーが発生していることを確認している。

4 インストール方法

ここではREVOCAP_Refinerのインストール方法、すなわちソルバに組み込むためのライブラリの作成手順を説明する。ソルバへ組み込むための API については、REVOCAP_Refiner 関数API一覧を参照のこと。組み込み方の具体的な方法はチュートリアルガイドを参照のこと。ソルバとのリンクの方法は、それぞれのソルバのマニュアルの参照のこと。

4.1 ソースの展開

提供されている圧縮されたソースコードREVOCAP_Refiner-X.X.X.tgzを展開する。X.X.Xはバージョン番号である。GNU系のtarコマンドで展開する場合は以下のようになる。

```
$ tar xvfz REVOCAP_Refiner-X.X.X.tgz
```

4.2 OPTIONS ファイルの編集

REVOCAP_Refinerを展開したディレクトリにはOPTIONSファイルが存在する。そこではビルドのためのフラグの設定などを行う。

4.3 MakefileConfig.in の編集

環境に依存した情報（コンパイラの設定など）は、REVOCAP_Refiner を展開したディレクトリにあるMakefileConfig.inファイルを編集する。このファイルはMakefileに読み込まれるので、Makefile のフォーマットに従って、サンプルをビルドする環境に応じて書き換える。主要な環境（PC-Cluster、Fujitsu Primergy、HA8000、ES2、Keiなど）についての設定例が同梱されているので、それをコピーして利用してもよい。既定ではPC-Cluster上でのGNU系のコンパイラを利用する場合が有効になっている。

4.4 make の実行

展開したディレクトリで、以下のコマンドを実行する。

```
$ make Refiner
```

REVOCAP_Refinerに関するmakeターゲットでは、以下の処理を実行することができる。

- Refiner：REVOCAP_Refiner のライブラリと前処理ツールをビルドする
- RefinerSampleC：REVOCAP_Refiner の C 言語サンプルをビルドする
- RefinerSampleF：REVOCAP_Refiner の Fortran 言語サンプルをビルドする

- RefinerDoc：REVOCAP_Refiner のドキュメントを生成する。Doxygen がインストールされている必要がある。
- RefinerClean：REVOCAP_Refiner に関連する生成したファイルを削除する。

4.5 生成したファイルの確認とテスト

OPTIONS ファイルに記述した LIB 変数と ARCH 変数の下に生成されたライブラリがある。たとえば PC クラスタで Linux 環境の場合には Refiner/lib/x86_64-linux 以下に生成される。ソルバがこのライブラリを C 言語で利用する場合は展開したディレクトリの Refiner/rcapRefiner.h をインクルードすればよい（他のヘッダーファイルをインクルードする必要はない）。

4.6 サンプルプログラム概説

詳細はチュートリアルを参照のこと。

main.c

もっとも単純な細分プログラムとして、2 つの四面体からなるモデルを細分する C 言語のプログラム。

mainHexa.c

２つの六面体からなるモデルを細分する C 言語のプログラム。

mainHexa2.c

２つの六面体 2 次要素からなるモデルを細分する C 言語のプログラム。

mainMultiStage.c

細分を 2 回繰り返す C 言語のプログラム。

mainMultiType.c

要素タイプが混在しているモデルを細分する C 言語のプログラム。

mainMultiByType.c

要素タイプが混在しているモデルを要素タイプごとにまとめて細分する C 言語のプログラム。

mainNV.c

境界面上の点を整数値で識別したモデルを細分する C 言語のプログラム。

mainFittingRefine.c

CAD 曲面による形状補正を行って細分する C 言語のプログラム。

mainShapeFuncFitting.c

２次要素の形状関数で形状補正をおこなって細分する C 言語のプログラム。

main.F90

main.c の Fortran90 言語版。

mainMulti.F90

mainMultiType.c の Fortran90 言語版。

test.rcap.f

FrontFlow/blue の中から呼び出す場合の Fortran 言語のサンプルプログラム。

5　形状適合機能

REVOCAP_Refiner の形状適合機能とは、細分前の疎なメッシュにおいて、CAD の形状データと疎なメッシュの表面との間に対応関係を生成しておき（前処理）、解析モデルの細分の際の中間節点の生成の時に、幾何的な中点ではなく、CAD の形状データが与える曲面のパラメータを使って中点を求めるものである。

5.1　CAD 形状データによる形状適合のための前処理

CAD の形状データを用いて、解析モデルの要素細分を行ったときに形状適合を行うには、細分前に CAD の形状データと、解析モデルの節点の間の対応付けを行っておく必要がある。

前処理は REVOCAP_PrePost または REVOCAP_Refiner 付属の前処理ツールで行う。具体的な方法については後述する。

5.2　CAD 形状データによる細分

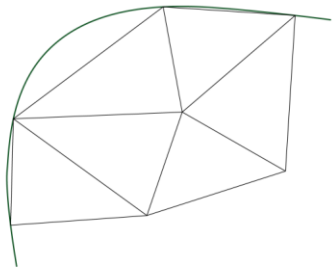


図 5-1　細分前

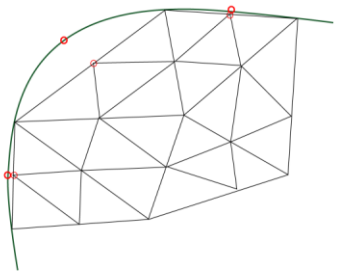


図 5-2　細分後

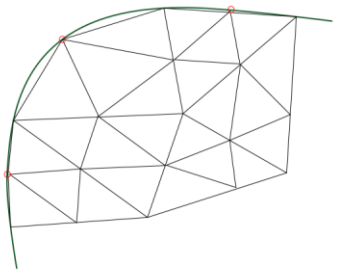


図 5-3 形状適合後

5.3 領域分割と形状補正の関係

REVOCAP_Refiner はソルバが並列計算を行う場合には領域分割後のモデルに対して要素の細分を行うが、領域分割後は計算ノードごとの局所節点番号でモデルが記述されているため、節点と CAD の形状データとの関係が与えられていない。そこで REVOCAP_Refiner があらかじめ計算ノードでの局所節点番号と大域的な節点番号を管理することで、局所節点番号と CAD の形状データとの関係を再構築する。それによって、領域分割後も形状補正が可能になる。

ソルバまたはカップラが大域節点番号と局所節点番号の対応を管理している場合は、ソルバまたはカップラが REVOCAP_Refiner を呼び出す場合にその対応を与えることで領域分割後の形状補正を行えるようにする。

実際に並列処理時に形状補正を行うためには以下の手順で行う。

(1) CAD データから疎なメッシュを生成するときに、曲面上にある節点と CAD 曲面の情報との対応をファイルに記録する（前処理ツールを利用）

(2) 疎なメッシュを領域分割し、節点の大域番号と局所番号の対応をファイルに記録する

(3) 領域分割後のそれぞれの計算ノードでソルバおよびカップラが REVOCAP_Refiner を呼び出す

(4) 各計算ノードで領域分割されたモデル（メッシュ）を読み込む

(5) 節点の大域番号と局所番号の対応を（それを記録したファイルを読み込んで）、REVOCAP_Refiner に与える

(6) 曲面上の節点の大域番号と CAD の曲面の対応を記録したファイルを読み込み、REVOCAP_Refiner に与える

(7) REVOCAP_Refiner の内部で自動的に曲面上にある領域分割後の節点の局所番号と CAD の曲面の対応関係を得る

(8) 要素の細分時にその要素の面が曲面上にあるときは、CAD の形状情報を使って、細分した面の節点を曲面上に乗るように REVOCAP_Refiner が補正をする

(9) 必要な場合は細分で生成した中間節点の座標を移動して、メッシュの品質の悪化を防ぐ

(10) 細分後に生成された中間節点を含め、曲面上にある節点の局所番号と CAD の曲面の対応情報をファイルに記録する

細分時に形状補正を行った場合に(10)で出力されるファイルは、節点の局所番号と CAD の曲面の対応が与えられている。従ってもう一度細分する場合に形状補正する場合には、大域節点番号を与えずに行えばよい。
すなわち 2 回目以降の細分時の形状補正、並列処理しない場合の細分時の形状補正は以下の手順で行う。

(1) 曲面上にある節点番号と CAD 曲面の情報との対応をファイルに記録する

(2) 領域分割後のそれぞれの計算ノードでソルバおよびカップラが REVOCAP_Refiner を呼び出す

(3) REVOCAP_Refiner にモデルを読み込む

(4) 曲面上の節点と CAD の曲面の対応を記録したファイルを読み込む

(5) 要素の細分時にその要素の面が曲面上にあるときは、CAD の形状情報を使って、細分した面の節点を曲面上に乗るように補正をする

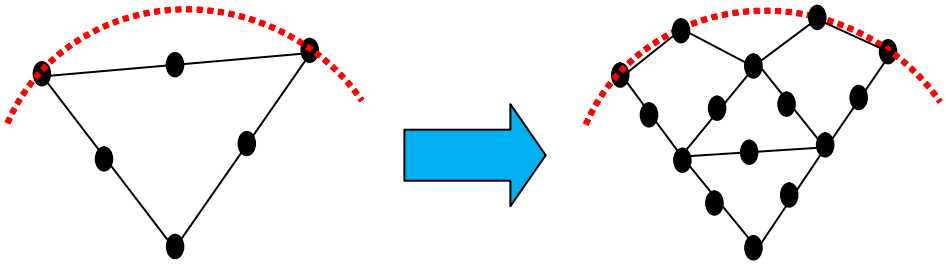
(6) 必要な場合は要素の細分の仕方を変えて、メッシュの品質の悪化を防ぐ

(7) 細分後に生成された中間節点を含め、曲面上にある節点の局所番号と CAD の曲面の対応情報を出力する

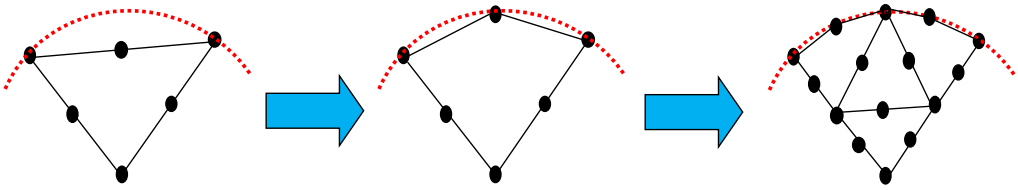
すなわち上記の(2) (5) (7)の工程を省略できる。
節点と CAD 曲面の対応関係を記録したファイルは、CAD ファイルフォーマットを参照のこと。

5.4 2 次要素の中間節点の形状適合について

形状適合されるメッシュが 2 次要素からなる場合、その中間節点が CAD 曲面上にない場合にそのまま形状適合を行って、新たに生成される中間節点だけを CAD 曲面上に移動させると、メッシュ形状がいびつな形になる。



この現象を回避するために、形状適合して細分を行う前にあらかじめ中間節点を CAD 曲面上に移動させる。



5.5　形状適合機能の制限事項

形状適合機能については以下の制限がある。

曲面データ構造について

REVOCAP_Refiner が扱うことのできる曲面のデータは CAD ファイルフォーマットで与えられる NURBS 形式に限ることに注意する。特に Primitive で表現されることの多い、単純な平面、円柱、円錐、球面なども同値な NURBS 表現に変換して与えなければならない。この変換は REVOCAP_PrePost でファイルを変換するときにデータ表現を変換することで行う。

また、単一の曲面で曲面座標(u,v)が周期的になっている場合、すなわち u の最小値と最大値で空間の同一の点を与えるような場合は REVOCAP_Refiner ではエラーが発生する場合がある。このような場合は、（多くの場合形式的に）いくつかの曲面に分ける必要がある。

曲面データをまたがる要素について

REVOCAP_Refiner に与えられるデータは曲面データをまたがる要素が存在する場合は、正しく形状適合ができない場合がある。例えば、円柱の側面や球面を複数の NURBS 面で表現したときに、それらの境界面をまたぐような要素が存在した場合は、その要素の中間節点をどの曲面データに適合させるかが決まらないため、形状適合が働かない場合がある。

6 CAD ファイルフォーマット

ここでは rcapSetCADFilename 関数で呼ばれる CAD ファイルのフォーマットについて記述する。
はじめに概要をまとめる。

- YAML フォーマットで記述する
- 記述可能な曲面の種類は、NURBS、B-Spline、Bezier の３種類である
- 同一のファイルにグローバル節点番号と曲面の間の対応関係も記述する
- このファイルは REVOCAP_PrePost または REVOCAP_Refiner の前処理ツールが生成することができる

一般的な YAML フォーマットの仕様ではインデントを使い階層構造を表現する。ただしインデントにはタブが使えずに、スペースのみが使えることに注意する。ここではスペース 2 個でインデントする。その他の YAML についての仕様は http://yaml.org/ 等を参照していただきたい。

6.1 トップレベルノード

曲面のコレクションを記述する surface、節点番号と曲面の対応関係を記述する data、およびその他の REVOCAP_Refiner のためのパラメータがある。

6.2 surface ノード

surface ノードの下位は nurbs、bspline、 bezier のいずれかからなる曲面のノードのコレクションである。

6.3 nurbs ノード

nurbs ノードは id、uknots、vknots、ctrlpts をキーとするマッピングである。

id		整数値	曲面を識別するための 0 以上の整数を与える。
uknots	order	整数値	nurbs 曲面の u 方向の位数を与える。
	vector	浮動小数点値のリスト	nurbs 曲面の u 方向のノットベクトルをあたえる。
vknots	order	整数値	nurbs 曲面の v 方

			向の位数を与える。
	vector	浮動小数点値のリスト	nurbs 曲面の v 方向のノットベクトルを与える。
ctrlPts		浮動小数点の 4 次元ベクトルのリスト	nurbs 曲面の制御点の (x,y,z) 座標とウェイトを合わせた 4 次元ベクトルのリストを与える。

6.4 data ノード

data ノードは name、mode、vtype、stype、size、value からなるマッピングのコレクションである。
これは REVOCAP_PrePost のニュートラルメッシュファイルのフォーマットと共通である。

name	文字列	他のデータと区別するための識別子。ここでは'fitting'という文字列を与える。
mode	文字列	データの種類を規定するための識別子。ここでは'NODEVARIABLE'という文字列を与える。
vtype	文字列	データの値を規定するための識別子。ここでは'VECTOR2WITHINT'という文字列を与える。
stype	文字列	このデータの付加的な意味を与えるための識別子。プログラム内部での処理のために使う。ここでは 'shape' という文字列を与える。
size	整数値	データの個数を与える。
value	整数値と（浮動小数点 2 個と整数値のベクトル）のペア	最初の整数値でグローバル節点番号、浮動小数点 2 個(u,v)と整数値は、整数値で曲面の id を与え、(u,v) で節

		点をその曲面上の最も近い点で与えたときの (u,v) 座標を与える。 節点番号に対して、複数のベクトルを与えてもよく、ベクトルが与えられていない節点が存在してもよい。 このファイルでは、ソルバーの節点番号の付け方にかかわらず、0 から始まる。

6.5 その他のパラメータ

トップレベルノードにその他の REVOCAP_Refiner のパラメータを記述することができる。

パラメータ名	内容	省略したときの値
epsilon	前処理における曲面と節点とのマッピング処理の Newton-Raphson 法の閾値	1.0e-8
itermax	前処理における曲面と節点とのマッピング処理の Newton-Raphson 法の最大反復回数	1000

6.6 例

例として、2 つ並んだ 6 面体の節点が円柱の表面で近似される場合を挙げる。

```
# 0.70710678 = sqrt(0.5)
# center (0.0, 0.0, 0.0)
# radius 1.0
# axis (1.0, 0.0, 0.0)
# height 2.0
---
surface:
  - nurbs:
      id: 0
      uknots:
        order: 3
```

```
        vector: [0.0, 0.0, 0.0, 0.5, 0.5, 1.0, 1.0, 1.0]
      vknots:
        order: 2
        vector: [0.0, 0.0, 1.0, 1.0]
      ctrlpts:
        - [0.0,  1.0,  0.0,  1.0]
        - [0.0,  1.0,  1.0,  0.70710678]
        - [0.0,  0.0,  1.0,  1.0]
        - [0.0, -1.0,  1.0,  0.70710678]
        - [0.0, -1.0,  0.0,  1.0]
        - [2.0,  1.0,  0.0,  1.0]
        - [2.0,  1.0,  1.0,  0.70710678]
        - [2.0,  0.0,  1.0,  1.0]
        - [2.0, -1.0,  1.0,  0.70710678]
        - [2.0, -1.0,  0.0,  1.0]
  - nurbs:
      id: 1
      uknots:
        order: 3
        vector: [0.0, 0.0, 0.0, 0.5, 0.5, 1.0, 1.0, 1.0]
      vknots:
        order: 2
        vector: [0.0, 0.0, 1.0, 1.0]
      ctrlpts:
        - [0.0, -1.0,  0.0,  1.0]
        - [0.0, -1.0, -1.0,  0.70710678]
        - [0.0,  0.0, -1.0,  1.0]
        - [0.0,  1.0, -1.0,  0.70710678]
        - [0.0,  1.0,  0.0,  1.0]
        - [2.0, -1.0,  0.0,  1.0]
        - [2.0, -1.0, -1.0,  0.70710678]
        - [2.0,  0.0, -1.0,  1.0]
        - [2.0,  1.0, -1.0,  0.70710678]
        - [2.0,  1.0,  0.0,  1.0]
data:
  - name: fitting
```

```
mode: NODEVARIABLE
vtype: VECTOR2WITHINT
stype: shape
size: 18
value:
- [0, [0.0, 0.0, 0]]
- [1, [0.5, 0.0, 0]]
- [2, [1.0, 0.0, 0]]
- [4, [0.0, 0.5, 0]]
- [5, [0.5, 0.5, 0]]
- [6, [1.0, 0.5, 0]]
- [8, [0.0, 1.0, 0]]
- [9, [0.5, 1.0, 0]]
- [10, [1.0, 1.0, 0]]
- [2, [0.0, 0.0, 1]]
- [3, [0.5, 0.0, 1]]
- [0, [1.0, 0.0, 1]]
- [6, [0.0, 0.5, 1]]
- [7, [0.5, 0.5, 1]]
- [4, [1.0, 0.5, 1]]
- [10, [0.0, 1.0, 1]]
- [11, [0.5, 1.0, 1]]
- [8, [1.0, 1.0, 1]]
```

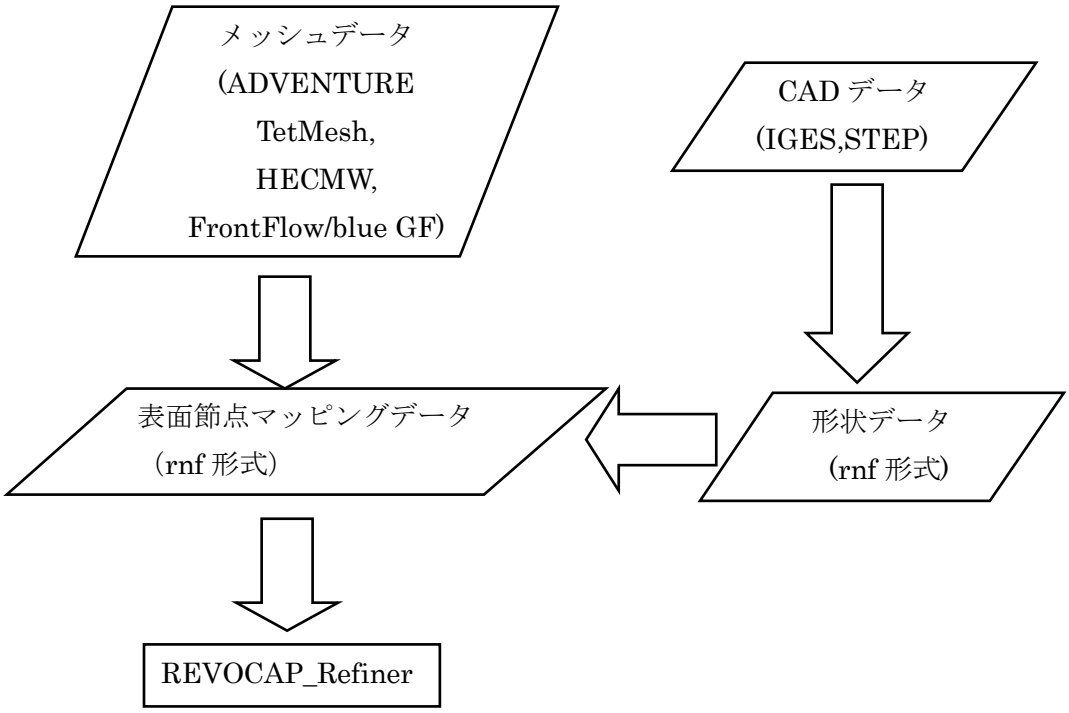
7 形状適合機能の前処理

「CAD ファイルフォーマット」で述べた rcapSetCADFilename 関数で呼ばれるファイルを作成する手順について説明する。大まかな手順は以下のとおりである。

(1) CAD ファイル（IGES 形式、STEP 形式）を REVOCAP_PrePost で読み込んで、独自形式（YAML 形式）の CAD ファイル（形状データ）に変換する。

(2) CAD ファイルに記述されている曲面と、細分の対象となるメッシュの表面の節点の間のマッチングを行う。マッチングの結果を data ノードに追加して出力する（表面節点マッピングデータ付きの YAML 形式の形状データ）。

(3) 出力された表面節点マッピングデータ付きの YAML 形式の形状データを rcapSetCADFilename で指定する。



上記(2)のプロセスは REVOCAP_PrePost で行うこともできるが、REVOCAP_Refiner 付属のツールで行うこともできる。

7.1 付属前処理ツールの作成

通常のライブラリをビルドする、make Refiner を実行したときに同時に前処理ツールのビルドが行われる。成功すれば、bin/{ARCH}/rcapPreFitting という実行ファイルが生成される。

前処理ツールのビルドだけを行いたい場合は、Refiner ディレクトリで make preFitting を

実行する。

7.2 前処理ツールの実行方法

ここでは前処理ツール rcapPreFitting の実行方法を説明する。引数は 3 つ指定して実行する。

```
$ rcapPreFitting originalmesh shapedatafile outputfile
```

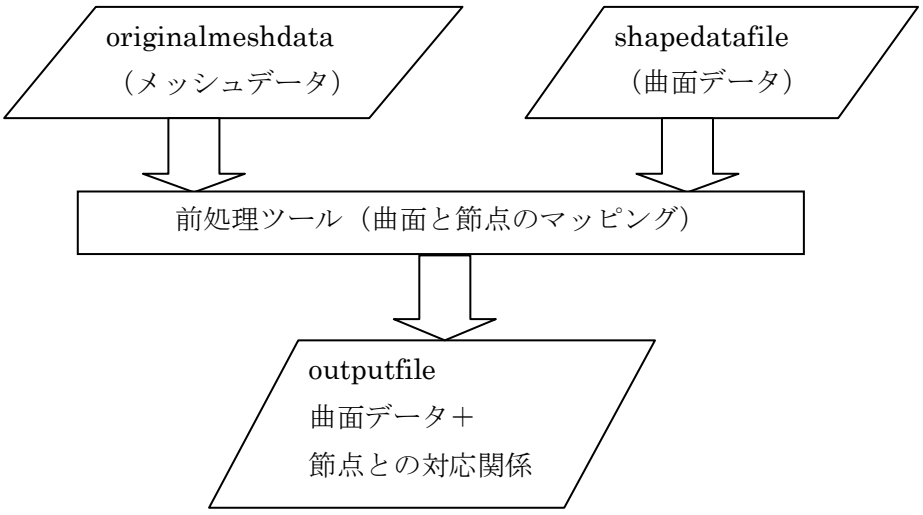
引数は順に

(1) *originalmesh* 細分前のメッシュファイル
(2) *shapedatafile* 入力 CAD ファイル（YAML 形式の形状データ）
(3) *outputfile* 出力 CAD ファイル（表面節点マッピングデータつきの YAML 形式の形状データ）

オプションとして、メッシュファイルのフォーマットを指定することができる。

- -a：ADVENTURE_TetMesh 形式（デフォルト）
- -h：HECMW（Ver.2）形式
- -b：FrontFlow/blue GF1 形式

rcapPreFitting を引数なしで実行した場合、簡単な使い方の説明が表示される。
outputfile は前処理ツールの出力で、REVOCAP_Refiner の入力データになる。前処理ツールは *shapedatafile* で与えられる曲面データに対して、節点と曲面の間の対応関係を生成するものである。*shapedatafile* は CAD ファイルフォーマットに従って記述されている必要があるが、data ノードは必要としない。*outputfile* は *shapedatafile* で与えられる曲面データに加えて data ノードが追加される。



8　形状関数による補正機能

2 次要素については、形状関数で与えられる要素内の座標が線形ではないので、要素内座標での中点と、幾何的な中点が異なる場合がある。ここでは前者の形状関数で中点を求めることで、境界を形状関数が表す曲面として補正する。
通常、中間節点が幾何的な中点と異なるものは境界面にだけ現れると考えられるため、ここでは要素の境界面上にある中間節点を生成するときに、境界面の要素の形状関数を使って補正を行う。有限要素法では、四面体の形状関数を境界面に制限したものが三角形の形状関数となるため、境界面に制限して計算することは妥当である。三角形 2 次要素の形状関数として、以下のものを採用した。

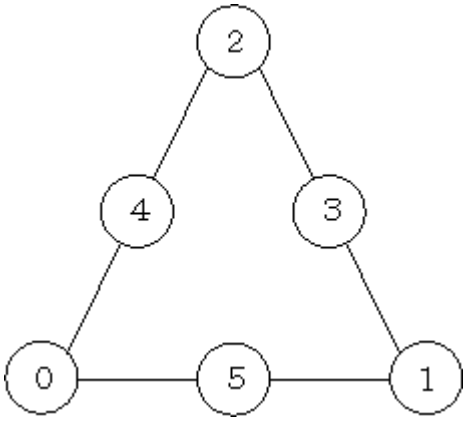


図 8-1 三角形 2 次要素

0	(1-s-t)*(1-2*s-2*t)
1	s*(2*s-1)
2	t*(2*t-1)
3	4*s*t
4	4*t*(1-s-t)
5	4*s*(1-s-t)

この 2 次要素を細分するときの要素内座標とそれぞれの形状関数の値をまとめると、次のようになる。

頂点番号	s	t	0	1	2	3	4	5
0と5	0.25	0	0.375	−0.125	0	0	0	0.75
5と1	0.75	0	−0.125	0.375	0	0	0	0.75
1と3	0.75	0.25	0	0.375	−0.125	0.75	0	0
3と2	0.25	0.75	0	−0.125	0.375	0.75	0	0
2と4	0	0.75	−0.125	0	0.375	0	0.75	0
4と0	0	0.25	0.375	0	−0.125	0	0.75	0
5と3	0.5	0.25	−0.125	0	−0.125	0.5	0.25	0.5
3と4	0.25	0.5	−0.125	−0.125	0	0.5	0.5	0.25
4と5	0.25	0.25	0	−0.125	−0.125	0.25	0.5	0.5

そこで次の3点関数と5点関数を内部で実装した。

createMiddleNode3(n0,n1,n2) = 0.375*n0 + 0.75*n1 -0.125*n2

createMiddleNode5(n0,n1,n2,n3,n4) = 0.5*n0 + 0.5*n1 -0.125*n2 – 0.125*n3 + 0.25*n4

三角形2次要素の辺上の中間節点の生成にはcreateMiddleNode3を用い、内部にある中間節点の生成にはcreateMiddleNode5を用いた。

9 メッシュ品質レポート機能

一般にメッシュの品質の指標とされているものとして

- アスペクト比
- 辺の長さの比
- 最大角度
- 最小角度

などが挙げられる。REVOCAP_Refiner では、rcapQualityReport 関数で、第一引数でメッシュの品質の指標名を文字列で与えて、レポートをファイルに出力する機能を実装している。レポート機能としては、それぞれの指標の最大値、最小値、平均値、簡易度数分布（ヒストグラム）を出力する。

10 メッシュ細分後品質改善機能

形状適合機能を利用した場合には、形状に適合するように表面の節点を移動させるために要素がいびつに変形してしまう場合がある。これを改善するための内部の節点を移動させることでメッシュの品質を改善する機能について説明する。

10.1 機能概要

本機能については以下のように品質を改善する

- ソルバーの境界条件、通信テーブルへの影響を考え、トポロジーを変えずに節点の座標だけを移動させる（品質改善ルーチンにおいて新たな要素の分割、節点の追加は行わない)。
- 表面上の節点（形状適合によって移動させた節点を含む）は動かさない。
- Refine 前の節点の座標は動かさない。
- 新しい節点の座標は Laplacian Smoothing で決める。
- 複数回の細分の場合、毎回の細分の後に品質補正を行う。

10.2 利用方法

rcapInitRefiner() で初期化した直後に、rcapSetSmoothing 関数で有効にするか無効にするかを設定し、その後に節点の設定、要素の細分を行えばよい。

付録: REVOCAP_Refiner 関数 API 一覧

void　rcapGetVersion (void)

Version 文字列を標準出力に出力する。

void　rcapInitRefiner (int32_t nodeOffset, int32_t elementOffset)

Refiner を初期化し、節点番号と要素番号のオフセット値を与える。

void　rcapClearRefiner (void)

Refiner が内部で保持している中間節点データのキャッシュをクリアする。

void　rcapTermRefiner (void)

Refiner の終了処理を行う。これを呼び出した後はすべての処理が無効になる。

void　rcapSetCADFilename (const char *filename)

形状補正に用いる CAD ファイルを指定する。
細分前のメッシュはこの CAD ファイルから生成されているものとする。 この関数を呼んで CAD ファイルを指定しなかった場合は、形状補正は行わない。 メッシュ生成時の節点の globalID と CAD の形状データの対応が与えられているとする。 領域分割後のメッシュに対して細分を行う場合は、setPartitionFilename などで globalID と localID の対応付けを与える必要がある。

void　rcapSetSecondFitting (int32_t flag)

中間節点の生成に 2 次要素の形状関数を使うかどうかを設定する。

void　rcapSetSmoothing (int32_t flag)

中間節点を Laplacian Smoothing するかどうかを設定する。

void　rcapSetPartitionFilename (const char *filename)

節点の globalID と localID の対応関係を記述したファイルを指定する。
指定しない場合は、globalID と localID は区別しない。 ファイルではなく、節点座標を登録するときに globalID と localID の関係を与えることもできる。

void　rcapSetNode64 (size_t num, float64_t *coords, int32_t *globalIds,

int32_t *localIds)

節点座標を Refiner に与える

void rcapSetNode32 (size_t num, float32_t *coords, int32_t *globalIds, int32_t *localIds)

節点座標を Refiner に与える

size_t rcapGetNodeCount (void)

現在 Refiner が保持している節点の個数を返す。細分したら増える。

void rcapGetNode64 (size_t num, int32_t *localIds, float64_t *coords)

Refiner が管理している節点座標を取得する

void rcapGetNode32 (size_t num, int32_t *localIds, float32_t *coords)

Refiner が管理している節点座標を取得する

void rcapGetNodeSeq64 (size_t num, size_t initId, float64_t *coords)

initId から連続して num 個の節点座標を取得する

void rcapGetNodeSeq32 (size_t num, size_t initId, float32_t *coords)

rcapGetNodeSeq64 の 32bit 版

size_t rcapRefineElement (size_t num, int8_t etype, int32_t *nodeArray, int32_t *resultNodeArray)

要素をそれぞれ辺の２分割して細分する

size_t rcapGetRefineElementCount (size_t num, int8_t etype)

細分した要素の個数を計算する（実際には細分はしない）

size_t rcapRefineElementMulti (size_t num, int8_t *etypeArray, int32_t *nodeArray, size_t *refinedNum, int8_t *resultEtypeArray, int32_t *resultNodeArray)

複数の種類の型が混在しているモデルを一度に細分する

size_t rcapGetRefineElementMultiCount (size_t num, int8_t *etypeArray, size_t *refinedNum)

複数の種類の型が混在しているモデルを細分したときの要素の個数を計算する（実際には細分しない）

void rcapCommit (void)

rcapRefineElement により細分されたデータ（節点グループ、要素グループ、面グループ）をコミットする。 すなわち、以下の rcapGet[Node | Element | Face]Group[Count] メソッドの対象を細分前のデータから 細分後のデータに変える。細分前のデータは削除される。この関数を実行後に rcapRefineElement を再度実行した場合、更新されるデータは 細分後のデータになる。 また、rcapRefineElement で細分後の要素に付与される要素番号も elementOffset 値にリセットされる。

void rcapAppendNodeGroup (const char dataname[80], size_t num, int32_t *nodeArray)

細分と同時に更新する節点グループを登録

size_t rcapGetNodeGroupCount (const char dataname[80])

Refiner に登録されている節点グループの節点の個数を返す

void rcapGetNodeGroup (const char dataname[80], size_t num, int32_t *nodeArray)

Refiner に登録されている節点グループを返す

void rcapAppendBNodeGroup (const char dataname[80], size_t num, int32_t *nodeArray)

BoundaryNodeGroup とは、境界面上にのみある節点グループのこと。この関数で登録する。

size_t rcapGetBNodeGroupCount (const char dataname[80])

Refiner に登録されている境界節点グループの節点の個数を返す

void rcapGetBNodeGroup (const char dataname[80], size_t num, int32_t *nodeArray)

Refiner に登録されている境界節点グループを返す

void rcapAppendBNodeVarInt (const char dataname[80], size_t num, int32_t *nodeArray, int32_t *nodeVars)

BoundaryNodeVariableInt とは、境界面上にのみある節点上の整数値変数のこと

size_t rcapGetBNodeVarIntCount (const char dataname[80])

Refiner に登録されている整数値境界節点変数の節点の個数を返す

void rcapGetBNodeVarInt (const char dataname[80], size_t num, int32_t *nodeArray, int32_t *nodeVars)

Refiner に登録されている整数値境界節点変数を返す

void rcapAppendElementGroup (const char dataname[80], size_t num, int32_t *elementArray)

ElementGroup とは、要素番号の集合のこと。

size_t rcapGetElementGroupCount (const char dataname[80])

Refiner に登録されている要素グループの要素の個数を返す

void rcapGetElementGroup (const char dataname[80], size_t num, int32_t *elementArray)

Refiner に登録されている要素グループを返す

void rcapAppendFaceGroup (const char dataname[80], size_t num, int32_t *faceArray)

FaceGroup とは、要素番号、要素内面番号の組のこと。連成面を細分する場合などに用いる。

size_t rcapGetFaceGroupCount (const char dataname[80])

Refiner に登録されている面グループの個数を返す

void rcapGetFaceGroup (const char dataname[80], size_t num, int32_t *faceArray)

Refiner に登録されている面グループを返す

void rcapSetInterpolateMode (const char mode[32])

NodeVariable を登録したときに、中間節点に与える値の決め方を選択します。 現在は

"MIN" "MAX" "MIDDLE" の 3 種類に対応しています。 MIN は中間節点を生成するのに用いた節点上の値の最小値を与えます。 MAX は中間節点を生成するのに用いた節点上の値の最大値を与えます。 MIDDLE は中間節点を生成するのに用いた節点上の値の平均値を与えます。

void rcapGetInterpolateMode (char mode[32])

NodeVariable を登録したときに、中間節点に与える値の決め方を返します。 戻り値は "MIN" "MAX" "MIDDLE" という文字列のいずれかです。

int8_t rcapGetOriginal (int32_t localNodeId, int32_t *originalNodeArray)

中間節点から、それを生成するのに使った辺、面、要素の節点配列を返す

int32_t rcapGetMiddle (int8_t etype, int32_t *originalNodeArray)

辺、面、要素を与えて、それから作られた中間節点を戻り値で返す

void rcapQualityReport (const char name[80], const char *filename)

要素の品質に関する情報を出力します。