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_Refiner Ver. 1.1

チュートリアルガイド

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 サンプルプログラム

1.1 四面体を細分する

ソースコードにいくつかのサンプルプログラムが添付されている。ここでは、最も単純な場合である四面体１つのモデルを細分する C 言語のプログラムを挙げる。

```
/*
 *
 * サンプル実行例＆テスト用プログラム
 * 四面体の細分チェック用
 *
 */

#ifdef _CONSOLE

#include "rcapRefiner.h"
#include "rcapRefinerMacros.h"
#include <stdio.h>
#include <stdlib.h>    /* for calloc, free */
```

```
#include <assert.h>

int main(void)
{
	/* 四面体を 1 つ並べる */
	float64_t coords[12] = {
		0.0, 0.0, 0.0,
		1.0, 0.0, 0.0,
		0.0, 1.0, 0.0,
		0.0, 0.0, 1.0,
	};
	size_t refineNodeCount = 0;
	float64_t* resultCoords = NULL;
	int32_t tetras[4] = {
		1, 2, 3, 4,
	};
	/* 細分後の四面体の節点配列：出力は 8 個 */
	int32_t* refineTetras = NULL;
	/* 細分する要素の型(定数値) */
	int8_t etype = RCAP_TETRAHEDRON;
	int32_t nodeOffset = 1;
	int32_t elementOffset = 1;
	/* 初期節点の個数 */
	size_t nodeCount = 4;
	/* 初期要素の個数 */
	size_t elementCount = 1;
	/* 細分後の要素の個数 */
	size_t refineElementCount = 0;

	/* 境界条件（節点グループ） */
	int32_t ng0[3] = {1,2,3};
	int32_t* result_ng0 = NULL;
	size_t ng0Count = 3;

	/* カウンタ */
	int32_t i,j;
```

```
	/* 節点番号のオフセット値を与える */
	rcapInitRefiner( nodeOffset, elementOffset );
	printf("----- Original Model -----¥n");
	/* 座標値を Refiner に与える */
	rcapSetNode64( nodeCount, coords, NULL, NULL );
	/* 細分前の節点数 */
	nodeCount = rcapGetNodeCount();
	assert( nodeCount == 4 );
	printf("Node：Count = %"PRIsz"¥n", nodeCount );
	for(i=0;(size_t)i<nodeCount;++i){
		printf("%d：%f, %f, %f¥n", i+nodeOffset, coords[3*i], coords[3*i+1],
coords[3*i+2] );
	}
	/* 細分前の要素数 */
	assert( elementCount == 1 );
	printf("Element：Count = %"PRIsz"¥n", elementCount );
	for(i=0;(size_t)i<elementCount;++i){
		printf("%d：(%d) %d, %d, %d, %d¥n", i+elementOffset, etype,
			tetras[4*i], tetras[4*i+1], tetras[4*i+2], tetras[4*i+3]);
	}
	/* 節点グループの登録 */
	rcapAppendNodeGroup("ng0",ng0Count,ng0);
	ng0Count = rcapGetNodeGroupCount("ng0");
	assert( ng0Count == 3 );
	printf("Node Group：Count = %"PRIsz"¥n", ng0Count );
	for(i=0;(size_t)i<ng0Count;++i){
		printf("%d¥n", ng0[i]);
	}

	printf("----- Refined Model -----¥n");

	/* 要素の細分 */
	refineElementCount = rcapRefineElement( elementCount, etype, tetras,
NULL );
	refineTetras = (int32_t*)calloc( 4*refineElementCount, sizeof(int32_t) );
```

```
	elementCount = rcapRefineElement( elementCount, etype, tetras,
refineTetras);
	rcapCommit();

	/* 細分後の節点 */
	refineNodeCount = rcapGetNodeCount();
	printf("Node : Count = %"PRIsz"¥n", refineNodeCount );
	resultCoords = (float64_t*)calloc( 3*refineNodeCount, sizeof(float64_t) );
	rcapGetNodeSeq64( refineNodeCount, nodeOffset, resultCoords );
	for(j=0;(size_t)j<refineNodeCount;++j){
		printf("%d : %f, %f, %f¥n", j+nodeOffset, resultCoords[3*j],
resultCoords[3*j+1], resultCoords[3*j+2] );
	}
	free( resultCoords );

	/* 細分後の要素 */
	printf("Element : Count = %"PRIsz"¥n", refineElementCount );
	for(i=0;(size_t)i<refineElementCount;++i){
		printf("%d : (%d) %d, %d, %d, %d¥n", i+elementOffset, etype,
			refineTetras[4*i], refineTetras[4*i+1], refineTetras[4*i+2],
refineTetras[4*i+3] );
	}

	/* 細分後の節点グループの更新 */
	ng0Count = rcapGetNodeGroupCount("ng0");
	result_ng0 = (int32_t*)calloc( ng0Count, sizeof(int32_t) );
	printf("Refined Node Group : Count = %"PRIsz"¥n", ng0Count );
	rcapGetNodeGroup("ng0",ng0Count,result_ng0);
	for(i=0;(size_t)i<ng0Count;++i){
		printf("%d¥n", result_ng0[i]);
	}
	free( result_ng0 );

	free( refineTetras );
	rcapTermRefiner();
	return 0;
```

```
}


#endif
```

1.2　六面体を細分する

```
/*
 *
 * サンプル実行例＆テスト用プログラム
 * 六面体の細分チェック用
 *
 */

#ifdef _CONSOLE

#include "rcapRefiner.h"
#include "rcapRefinerMacros.h"
#include <stdio.h>
#include <stdlib.h>    /* for calloc, free */
#include <assert.h>

int main(void)
{
        /* 六面体を 2 つ並べる */
        float64_t coords[36] = {
                0.0, 0.0, -1.0,
                1.0, 0.0, -1.0,
                1.0, 1.0, -1.0,
                0.0, 1.0, -1.0,
                0.0, 0.0,  0.0,
                1.0, 0.0,  0.0,
                1.0, 1.0,  0.0,
                0.0, 1.0,  0.0,
                0.0, 0.0,  1.0,
                1.0, 0.0,  1.0,
                1.0, 1.0,  1.0,
                0.0, 1.0,  1.0,
        };
        size_t refineNodeCount = 0;
        float64_t* resultCoords = NULL;
        int32_t hexas[16] = {
```

```
            1, 2, 3, 4, 5, 6, 7, 8,
            5, 6, 7, 8, 9,10,11,12
};
/* 細分後の六面体の節点配列：出力は 2*8=16 個 */
int32_t* refineHexas = NULL;
/* 細分する要素の型(定数値) */
int8_t etype = RCAP_HEXAHEDRON;
int32_t nodeOffset = 1;
int32_t elementOffset = 1;
/* 初期節点の個数 */
size_t nodeCount = 12;
/* 初期要素の個数 */
size_t elementCount = 2;
/* 細分後の要素の個数 */
size_t refineElementCount = 0;

/* 境界条件（節点グループ） */
int32_t ng0[4] = {1,2,5,6};
int32_t* result_ng0 = NULL;
size_t ng0Count = 4;
/* 境界条件（境界節点グループ） */
int32_t bng0[3] = {5,6,7};
int32_t* result_bng0 = NULL;
size_t bng0Count = 3;

/* カウンタ */
int32_t i,j;

/* 節点番号のオフセット値を与える */
rcapInitRefiner( nodeOffset, elementOffset );
printf("----- Original Model -----¥n");
/* 座標値を Refiner に与える */
rcapSetNode64( nodeCount, coords, NULL, NULL );
/* 細分前の節点数 */
nodeCount = rcapGetNodeCount();
assert( nodeCount == 12 );
```

```
	printf("Node : Count = %"PRIsz"¥n", nodeCount );
	for(i=0;(size_t)i<nodeCount;++i){
		printf("%d : %f, %f, %f¥n", i+nodeOffset, coords[3*i], coords[3*i+1],
coords[3*i+2] );
	}
	/* 細分前の要素数 */
	assert( elementCount == 2 );
	printf("Element : Count = %"PRIsz"¥n", elementCount );
	for(i=0;(size_t)i<elementCount;++i){
		printf("%d : (%d) %d, %d, %d, %d, %d, %d, %d, %d¥n",
i+elementOffset, etype,
			hexas[8*i], hexas[8*i+1], hexas[8*i+2], hexas[8*i+3],
			hexas[8*i+4], hexas[8*i+5], hexas[8*i+6], hexas[8*i+7] );
	}
	/* 節点グループの登録 */
	rcapAppendNodeGroup("ng0",ng0Count,ng0);
	ng0Count = rcapGetNodeGroupCount("ng0");
	assert( ng0Count == 4 );
	printf("Node Group : Count = %"PRIsz"¥n", ng0Count );
	for(i=0;(size_t)i<ng0Count;++i){
		printf("%d¥n", ng0[i]);
	}
	rcapAppendBNodeGroup("bng0",bng0Count,bng0);
	bng0Count = rcapGetBNodeGroupCount("bng0");
	assert( bng0Count == 3 );
	printf("Boundary Node Group : Count = %"PRIsz"¥n", bng0Count );
	for(i=0;(size_t)i<bng0Count;++i){
		printf("%d¥n", bng0[i]);
	}

	printf("----- Refined Model -----¥n");

	/* 要素の細分 */
	refineElementCount = rcapRefineElement( elementCount, etype, hexas,
NULL );
	refineHexas = (int32_t*)calloc( 8*refineElementCount, sizeof(int32_t) );
```

```
	elementCount = rcapRefineElement( elementCount, etype, hexas, refineHexas );
	rcapCommit();

	/* 細分後の節点 */
	refineNodeCount = rcapGetNodeCount();
	printf("Node：Count = %"PRIsz"¥n", refineNodeCount );
	resultCoords = (float64_t*)calloc( 3*refineNodeCount, sizeof(float64_t) );
	rcapGetNodeSeq64( refineNodeCount, nodeOffset, resultCoords );
	for(j=0;(size_t)j<refineNodeCount;++j){
		printf("%d : %f, %f, %f¥n", j+nodeOffset, resultCoords[3*j], resultCoords[3*j+1], resultCoords[3*j+2] );
	}
	free( resultCoords );

	/* 細分後の要素 */
	printf("Element：Count = %"PRIsz"¥n", refineElementCount );
	for(i=0;(size_t)i<refineElementCount;++i){
		printf("%d : (%d) %d, %d, %d, %d, %d, %d, %d, %d¥n", i+elementOffset, etype,
			refineHexas[8*i], refineHexas[8*i+1], refineHexas[8*i+2], refineHexas[8*i+3],
			refineHexas[8*i+4], refineHexas[8*i+5], refineHexas[8*i+6], refineHexas[8*i+7] );
	}

	/* 細分後の節点グループの更新 */
	ng0Count = rcapGetNodeGroupCount("ng0");
	result_ng0 = (int32_t*)calloc( ng0Count, sizeof(int32_t) );
	printf("Refined Node Group：Count = %"PRIsz"¥n", ng0Count );
	rcapGetNodeGroup("ng0",ng0Count,result_ng0);
	for(i=0;(size_t)i<ng0Count;++i){
		printf("%d¥n", result_ng0[i]);
	}
	free( result_ng0 );
```

```
	bng0Count = rcapGetBNodeGroupCount("bng0");
	result_bng0 = (int32_t*)calloc( bng0Count, sizeof(int32_t) );
	printf("Refined Boundary Node Group : Count = %"PRIsz"¥n", bng0Count );
	rcapGetBNodeGroup("bng0",bng0Count,result_bng0);
	for(i=0;(size_t)i<bng0Count;++i){
		printf("%d¥n", result_bng0[i]);
	}
	free( result_bng0 );

	free( refineHexas );

	rcapTermRefiner();
	return 0;
}

#endif
```

1.3　多段細分する

```
/*
 *
 *  2 段階実行サンプルプログラム
 *
 */

#ifdef _CONSOLE

#include "rcapRefiner.h"
#include "rcapRefinerMacros.h"
#include <stdio.h>
#include <stdlib.h>    /* for calloc, free */
#include <assert.h>

int main(void)
{
        /*
         *  使い方の例
         *  初めの 5 つは細分する前の節点座標
         */
        float64_t coords[15] = {
                0.0, 0.0, 0.0,
                1.0, 0.0, 0.0,
                0.0, 1.0, 0.0,
                0.0, 0.0, 1.0,
                1.0, 1.0, 1.0
        };
        /*  細分後の座標：必要に応じて  calloc  する  */
        float64_t* resultCoords = NULL;
        /*  四面体の節点配列：入力は 2 個  */
        int32_t tetras[8] = {
                0, 1, 2, 3,
                1, 2, 3, 4
        };
        /*  細分後の四面体の節点配列：出力は 2*8=16 個*/
```

```
int32_t* refineTetras = NULL;
/*  ２段階細分後の四面体の節点配列：出力は 2*8*8=128 個*/
int32_t* refine2Tetras = NULL;
/*  細分する要素の型(定数値) */
int8_t etype = RCAP_TETRAHEDRON;
/*  メッシュの節点配列に現れる節点番号の最初の値  */
/* C  から呼ぶときは  0 fortran  から呼ぶ場合は  1 */
int32_t nodeOffset = 0;
int32_t elementOffset = 0;
/*  初期節点の個数  */
size_t nodeCount = 5;
/*  初期要素の個数  */
size_t elementCount = 2;
/*  細分後の要素の個数  */
size_t refineElementCount = 0;
/*  細分後の節点の個数  */
size_t refineNodeCount = 0;
/*  要素の細分と同時に更新する節点グループ  */
int32_t ng0[3] = {0,1,4};
size_t ngCount = 3;
int32_t* result_ng0 = NULL;
/*  要素の細分と同時に更新する面グループ  */
/*  要素番号と要素内面番号の順に交互に並べる  */
int32_t fg0[4] = {0,0,1,1};   /* [1,2,3] [1,4,3] */
size_t fgCount = 2;
int32_t* result_fg0 = NULL;

/*  ループのカウンタ  */
int32_t i = 0;
int32_t j = 0;

/*  節点番号のオフセット値を与える  */
rcapInitRefiner( nodeOffset, elementOffset );

printf("----- Original Model -----¥n");
/*
```

```
        * globalId と座標値を Refiner に教える
        * localIds は NULL をあたえると coords は nodeOffset から順番に並んでいるものと解釈する
        */
        rcapSetNode64( nodeCount, coords, NULL, NULL );
        /* 細分前の節点数 */
        nodeCount = rcapGetNodeCount();
        assert( nodeCount == 5 );
        printf("Node：Count = %"PRIsz"¥n", nodeCount );
        for(i=0;(size_t)i<nodeCount;++i){
                printf("%d：%f, %f, %f¥n", i+nodeOffset, coords[3*i], coords[3*i+1], coords[3*i+2] );
        }
        /* 細分前の要素数 */
        assert( elementCount == 2 );
        printf("Element：Count = %"PRIsz"¥n", elementCount );
        for(i=0;i<2;++i){
                printf("%d ： (%d) %d, %d, %d, %d¥n", i+elementOffset, etype, tetras[4*i], tetras[4*i+1], tetras[4*i+2], tetras[4*i+3] );
        }
        /* 節点グループの登録 */
        rcapAppendNodeGroup("innovate",ngCount,ng0);
        ngCount = rcapGetNodeGroupCount("innovate");
        assert( ngCount == 3 );
        printf("Node Group：Count = %"PRIsz"¥n", ngCount );
        for(i=0;i<3;++i){
                printf("%d¥n", ng0[i]);
        }
        /* 面グループの登録 */
        rcapAppendFaceGroup("revolute",fgCount,fg0);
        fgCount = rcapGetFaceGroupCount("revolute");
        assert( fgCount == 2 );
        printf("Face Group：Count = %"PRIsz"¥n", fgCount );
        for(i=0;i<2;++i){
                printf("%d, %d¥n", fg0[2*i], fg0[2*i+1]);
        }
```

```
	/*---------------------- REFINE STEP 1 ----------------------------------------*/

	/* 要素の細分 */
	refineElementCount = rcapRefineElement( elementCount, etype, tetras,
NULL );
	assert( refineElementCount == 16 );
	refineTetras = (int32_t*)calloc( 4*refineElementCount, sizeof(int32_t) );
	refineElementCount = rcapRefineElement( elementCount, etype, tetras,
refineTetras );
	assert( refineElementCount == 16 );
	rcapCommit();

	printf("----- Refined Model 1 -----¥n");

	/* 細分後の節点 */
	refineNodeCount = rcapGetNodeCount();
	printf("Node：Count = %"PRIsz"¥n", refineNodeCount );

	resultCoords = (float64_t*)calloc( 3*refineNodeCount, sizeof(float64_t) );
	rcapGetNodeSeq64( refineNodeCount, nodeOffset, resultCoords );
	for(j=0;(size_t)j<refineNodeCount;++j){
		printf("%d : %f, %f, %f¥n", j+nodeOffset, resultCoords[3*j],
resultCoords[3*j+1], resultCoords[3*j+2] );
	}

	/* 細分後の要素 */
	printf("Element：Count = %"PRIsz"¥n", refineElementCount );
	for(i=0;(size_t)i<refineElementCount;++i){
		printf("%d : (%d) %d, %d, %d, %d¥n", i+elementOffset, etype,
refineTetras[4*i], refineTetras[4*i+1], refineTetras[4*i+2], refineTetras[4*i+3] );
	}

	/* 細分後の節点グループの更新 */
	ngCount = rcapGetNodeGroupCount("innovate");
	printf("Node Group：Count = %"PRIsz"¥n", ngCount );
```

```
	assert( ngCount > 0 );
	result_ng0 = (int32_t*)calloc( ngCount, sizeof(int32_t) );
	rcapGetNodeGroup("innovate",ngCount,result_ng0);
	for(i=0;(size_t)i<ngCount;++i){
		printf("%d¥n", result_ng0[i]);
	}
	free( result_ng0 );
	result_ng0 = NULL;

	/* 細分後の面グループの更新 */
	fgCount = rcapGetFaceGroupCount("revolute");
	printf("Face Group：Count = %"PRIsz"¥n", fgCount );
	assert( fgCount > 0 );
	result_fg0 = (int32_t*)calloc( fgCount*2, sizeof(int32_t) );
	rcapGetFaceGroup("revolute",fgCount,result_fg0);
	assert( fgCount == 8 );
	for(i=0;(size_t)i<fgCount;++i){
		printf("%d, %d¥n", result_fg0[2*i], result_fg0[2*i+1]);
	}
	free( result_fg0 );
	result_fg0 = NULL;

	free( resultCoords );
	resultCoords = NULL;

	/* 第２段の細分の前にキャッシュをクリア */
	rcapClearRefiner();

	/*---------------------- REFINE STEP 2 ----------------------------------------*/

	/* 要素の細分 */
	elementCount = refineElementCount;
	refineElementCount = rcapRefineElement( elementCount, etype,
refineTetras, NULL );
	refine2Tetras = (int32_t*)calloc( 4*refineElementCount, sizeof(int32_t) );
	refineElementCount = rcapRefineElement( elementCount, etype,
```

```
refineTetras, refine2Tetras );
        rcapCommit();

        printf("----- Refined Model 2 -----¥n");

        /* 細分後の節点 */
        refineNodeCount = rcapGetNodeCount();
        printf("Node : Count = %"PRIsz"¥n", refineNodeCount );

        resultCoords = (float64_t*)calloc( 3*refineNodeCount, sizeof(float64_t) );
        rcapGetNodeSeq64( refineNodeCount, nodeOffset, resultCoords );
        for(j=0;(size_t)j<refineNodeCount;++j){
                printf("%d : %f, %f, %f¥n", j+nodeOffset, resultCoords[3*j],
resultCoords[3*j+1], resultCoords[3*j+2] );
        }

        /* 細分後の要素 */
        printf("Element : Count = %"PRIsz"¥n", refineElementCount );
        for(i=0;(size_t)i<refineElementCount;++i){
                printf("%d : (%d) %d, %d, %d, %d¥n", i+elementOffset, etype,
                        refine2Tetras[4*i], refine2Tetras[4*i+1],
refine2Tetras[4*i+2], refine2Tetras[4*i+3] );
        }

        /* 細分後の節点グループの更新 */
        ngCount = rcapGetNodeGroupCount("innovate");
        printf("Node Group : Count = %"PRIsz"¥n", ngCount );
        assert( ngCount > 0 );
        result_ng0 = (int32_t*)calloc( ngCount, sizeof(int32_t) );
        rcapGetNodeGroup("innovate",ngCount,result_ng0);
        for(i=0;(size_t)i<ngCount;++i){
                printf("%d¥n", result_ng0[i]);
        }
        free( result_ng0 );
        result_ng0 = NULL;
```

```
	/* 細分後の面グループの更新 */
	fgCount = rcapGetFaceGroupCount("revolute");
	printf("Face Group：Count = %"PRIsz"¥n", fgCount );
	assert( fgCount > 0 );
	result_fg0 = (int32_t*)calloc( fgCount*2, sizeof(int32_t) );
	rcapGetFaceGroup("revolute",fgCount,result_fg0);
	assert( fgCount == 8 );
	for(i=0;(size_t)i<fgCount;++i){
		printf("%d, %d¥n", result_fg0[2*i], result_fg0[2*i+1]);
	}
	free( result_fg0 );
	result_fg0 = NULL;

	free( resultCoords );
	resultCoords = NULL;

	free( refineTetras );
	refineTetras = NULL;
	free( refine2Tetras );
	refine2Tetras = NULL;

	rcapTermRefiner();
	return 0;
}

#endif
```

1.4　混合要素を細分する

```
/*
 *
 * サンプル実行例＆テスト用プログラム
 * 複数種類の要素細分チェック用
 *
 */

#ifdef _CONSOLE

#include "rcapRefiner.h"
#include "rcapRefinerMacros.h"
#include <stdio.h>
#include <stdlib.h>    /* for calloc, free */
#include <assert.h>

int main(void)
{
        /* 六面体の上に三角柱を乗せる */
        float64_t coords[30] = {
                0.0, 0.0, -1.0,
                1.0, 0.0, -1.0,
                1.0, 1.0, -1.0,
                0.0, 1.0, -1.0,
                0.0, 0.0,  0.0,
                1.0, 0.0,  0.0,
                1.0, 1.0,  0.0,
                0.0, 1.0,  0.0,
                0.5, 0.0,  1.0,
                0.5, 1.0,  1.0,
        };
        size_t refineNodeCount = 0;
        float64_t* resultCoords = NULL;
        int32_t hexas[8] = {
                1, 2, 3, 4, 5, 6, 7, 8,
        };
```

```
int32_t wedges[6] = {
        5, 9, 6, 8, 10, 7,
};
/* 細分後の六面体の節点配列：出力は 1*8=8 個 */
int32_t* refineHexas = NULL;
/* 細分後の三角柱の節点配列：出力は 1*8=8 個 */
int32_t* refineWedges = NULL;
/* 細分する要素の型(定数値) */
int8_t etype = RCAP_HEXAHEDRON;
int32_t nodeOffset = 1;
int32_t elementOffset = 1;
/* 初期節点の個数 */
size_t nodeCount = 10;
/* 初期要素の個数 */
size_t elementCount = 2;
/* 細分後の要素の個数 */
size_t refineHexaCount = 0;
size_t refineWedgeCount = 0;
size_t refineElementCount = 0;

/* 境界条件（節点グループ） */
int32_t ng0[5] = {1,2,5,6,9};
int32_t* result_ng0 = NULL;
size_t ng0Count = 5;

/* カウンタ */
int32_t i,j;

/* 節点番号のオフセット値を与える */
rcapInitRefiner( nodeOffset, elementOffset );
/* 座標値を Refiner に与える */
rcapSetNode64( nodeCount, coords, NULL, NULL );

printf("----- Original Model -----¥n");
/* 細分前の節点数 */
nodeCount = rcapGetNodeCount();
```

```
	assert( nodeCount == 10 );
	printf("Node : Count = %"PRIsz"¥n", nodeCount );
	for(i=0;(size_t)i<nodeCount;++i){
		printf("%d : %f, %f, %f¥n", i+nodeOffset, coords[3*i], coords[3*i+1],
coords[3*i+2] );
	}
	/* 細分前の要素数 */
	assert( elementCount == 2 );
	printf("Element : Count = %"PRIsz"¥n", elementCount );
	j = 0;
	etype = RCAP_HEXAHEDRON;
	elementCount = 1;
	for(i=0;(size_t)i<elementCount;++i){
		printf("%d : (%d) %d, %d, %d, %d, %d, %d, %d, %d¥n",
j+elementOffset, etype,
			hexas[8*i], hexas[8*i+1], hexas[8*i+2], hexas[8*i+3],
			hexas[8*i+4], hexas[8*i+5], hexas[8*i+6], hexas[8*i+7] );
		j++;
	}
	etype = RCAP_WEDGE;
	elementCount = 1;
	for(i=0;(size_t)i<elementCount;++i){
		printf("%d : (%d) %d, %d, %d, %d, %d, %d¥n", j+elementOffset,
etype,
			wedges[6*i], wedges[6*i+1], wedges[6*i+2],
			wedges[6*i+3], wedges[6*i+4], wedges[6*i+5] );
		j++;
	}

	/* 節点グループの登録 */
	rcapAppendNodeGroup("ng0",ng0Count,ng0);
	ng0Count = rcapGetNodeGroupCount("ng0");
	printf("Node Group : Count = %"PRIsz"¥n", ng0Count );
	assert( ng0Count == 5 );
	for(i=0;(size_t)i<ng0Count;++i){
		printf("%d¥n", ng0[i]);
```

```
	}

	printf("----- Refined Model -----¥n");
	/* 要素の細分 */
	etype = RCAP_HEXAHEDRON;
	elementCount = 1;
	refineHexaCount = rcapRefineElement( elementCount, etype, hexas, NULL );
	refineHexas = (int32_t*)calloc( 8*refineHexaCount, sizeof(int32_t) );
	etype = RCAP_WEDGE;
	elementCount = 1;
	refineWedgeCount = rcapRefineElement( elementCount, etype, wedges, NULL );
	refineWedges = (int32_t*)calloc( 6*refineWedgeCount, sizeof(int32_t) );
	refineElementCount = 0;
	etype = RCAP_HEXAHEDRON;
	elementCount = 1;
	refineHexaCount = rcapRefineElement( elementCount, etype, hexas, refineHexas );
	refineElementCount += refineHexaCount;
	etype = RCAP_WEDGE;
	elementCount = 1;
	refineWedgeCount = rcapRefineElement( elementCount, etype, wedges, refineWedges );
	refineElementCount += refineWedgeCount;
	rcapCommit();

	/* 細分後の節点 */
	refineNodeCount = rcapGetNodeCount();
	printf("Node：Count = %"PRIsz"¥n", refineNodeCount );
	resultCoords = (float64_t*)calloc( 3*refineNodeCount, sizeof(float64_t) );
	rcapGetNodeSeq64( refineNodeCount, nodeOffset, resultCoords );
	for(j=0;(size_t)j<refineNodeCount;++j){
		printf("%d : %f, %f, %f¥n", j+nodeOffset, resultCoords[3*j], resultCoords[3*j+1], resultCoords[3*j+2] );
	}
```

```
	free( resultCoords );

	/* 細分後の要素 */
	printf("Element : Count = %"PRIsz"¥n", refineElementCount );
	j = 0;
	etype = RCAP_HEXAHEDRON;
	for(i=0;(size_t)i<refineHexaCount;++i){
		printf("%d : (%d) %d, %d, %d, %d, %d, %d, %d, %d¥n",
j+elementOffset, etype,
			refineHexas[8*i], refineHexas[8*i+1], refineHexas[8*i+2],
refineHexas[8*i+3],
			refineHexas[8*i+4], refineHexas[8*i+5],
refineHexas[8*i+6], refineHexas[8*i+7] );
		j++;
	}
	etype = RCAP_WEDGE;
	for(i=0;(size_t)i<refineWedgeCount;++i){
		printf("%d : (%d) %d, %d, %d, %d, %d, %d¥n", j+elementOffset,
etype,
			refineWedges[6*i], refineWedges[6*i+1],
refineWedges[6*i+2],
			refineWedges[6*i+3], refineWedges[6*i+4],
refineWedges[6*i+5] );
		j++;
	}

	/* 細分後の節点グループ */
	ng0Count = rcapGetNodeGroupCount("ng0");
	result_ng0 = (int32_t*)calloc( ng0Count, sizeof(int32_t) );
	printf("Node Group : Count = %"PRIsz"¥n", ng0Count );
	rcapGetNodeGroup("ng0",ng0Count,result_ng0);
	for(j=0;(size_t)j<ng0Count;++j){
		printf("%d¥n", result_ng0[j]);
	}
	free( result_ng0 );
```

```
        free( refineHexas );
        free( refineWedges );

        rcapTermRefiner();
        return 0;
}

#endif
```

1.5　細分時に形状補正を行う

実行するディレクトリから相対パスで data/column2/column2.rnf に形状データと曲面と表面節点の対応を記述したファイルを置くこと。利用者の便宜のため、Refiner/data/column2 以下に columns2.rnf および、このサンプルと同じ六面体 2 つからメッシュを HECMW 形式（hec_column.msh）と FrontFlow/blue GF 形式（ffb_column.mesh）で同梱してあるので、適宜利用されたい。

```
/*
 *
 * 形状補正機能テスト用プログラムサンプル
 * 六面体２つからなる格子を円柱で形状補正する
 *
 * Sample Program for refinement with fitting to CAD surfaces.
 * This model consists two hexahedra with a column.
 *
 */

#ifdef _CONSOLE

#include "rcapRefiner.h"
#include "rcapRefinerMacros.h"
#include <stdio.h>
#include <stdlib.h>    /* for calloc, free */
#include <assert.h>

int main(void)
{
        int32_t nodeOffset = 1;
        int32_t elementOffset = 1;
        float64_t coords[36] = {
                0.0, 1.0, 0.0,
                0.0, 0.0, 1.0,
                0.0,-1.0, 0.0,
                0.0, 0.0,-1.0,
                1.0, 1.0, 0.0,
                1.0, 0.0, 1.0,
```

```
            1.0,-1.0, 0.0,
            1.0, 0.0,-1.0,
            2.0, 1.0, 0.0,
            2.0, 0.0, 1.0,
            2.0,-1.0, 0.0,
            2.0, 0.0,-1.0,
};
// CAD ファイルに記述してあるグローバル節点番号との対応
// nodeOffset = 1 を引いた値が CAD ファイルに記述してあることに注意
int32_t globalIds[12] = {
            101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112
};
int32_t localIds[12] = {
            1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12
};
size_t refineNodeCount = 0;
float64_t* resultCoords = NULL;
int32_t hexas[16] = {
            1, 2, 3, 4, 5, 6, 7, 8,
            5, 6, 7, 8, 9, 10, 11, 12,
};
/* 細分後の要素の節点配列 */
int32_t* refineHexas = NULL;
int8_t etype = RCAP_HEXAHEDRON;
/* 初期節点の個数 */
size_t nodeCount = 12;
/* 初期要素の個数 */
size_t elementCount = 2;
/* 細分後の要素の個数 */
size_t refineElementCount = 0;

/* 境界条件（節点グループ） */
int32_t ng0[4] = {1,4,5,8};
int32_t* result_ng0 = NULL;
size_t ng0Count = 4;
```

```
	/* カウンタ */
	int32_t i,j;

	/* 節点番号のオフセット値を与える */
	rcapInitRefiner( nodeOffset, elementOffset );
	rcapSetCADFilename( "data/column2/column2.rnf" );

	printf("----- Original Model -----¥n");
	/* 座標値を Refiner に与える */
	/* 節点配列で与える局所節点番号と、CAD ファイルに記述してある大域節点番号との対応も与える */
	rcapSetNode64( nodeCount, coords, globalIds, localIds );
	/* 細分前の節点数 */
	nodeCount = rcapGetNodeCount();
	assert( nodeCount == 12 );
	printf("Node：Count = %"PRIsz"¥n", nodeCount );
	for(i=0;(size_t)i<nodeCount;++i){
		printf("%d：%f, %f, %f¥n", i+nodeOffset, coords[3*i], coords[3*i+1], coords[3*i+2] );
	}
	/* 細分前の要素数 */
	assert( elementCount == 2 );
	printf("Element：Count = %"PRIsz"¥n", elementCount );
	for(i=0;(size_t)i<elementCount;++i){
		printf("%d : (%d) %d, %d, %d, %d, %d, %d, %d, %d¥n", i+elementOffset, etype,
			hexas[8*i], hexas[8*i+1], hexas[8*i+2], hexas[8*i+3],
			hexas[8*i+4], hexas[8*i+5], hexas[8*i+6], hexas[8*i+7]);
	}
	/* 節点グループの登録 */
	rcapAppendNodeGroup("ng0",ng0Count,ng0);
	ng0Count = rcapGetNodeGroupCount("ng0");
	assert( ng0Count == 4 );
	printf("Node Group：Count = %"PRIsz"¥n", ng0Count );
	for(i=0;(size_t)i<ng0Count;++i){
		printf("%d¥n", ng0[i]);
```

```
	}

	printf("----- Refined Model -----¥n");

	/* 要素の細分 */
	refineElementCount = rcapRefineElement( elementCount, etype, hexas, NULL );
	refineHexas = (int32_t*)calloc( 8*refineElementCount, sizeof(int32_t) );
	elementCount = rcapRefineElement( elementCount, etype, hexas, refineHexas);
	rcapCommit();

	/* 細分後の節点 */
	refineNodeCount = rcapGetNodeCount();
	printf("Node : Count = %"PRIsz"¥n", refineNodeCount );
	resultCoords = (float64_t*)calloc( 3*refineNodeCount, sizeof(float64_t) );
	rcapGetNodeSeq64( refineNodeCount, nodeOffset, resultCoords );
	for(j=0;(size_t)j<refineNodeCount;++j){
		printf("%d : %f, %f, %f¥n", j+nodeOffset, resultCoords[3*j], resultCoords[3*j+1], resultCoords[3*j+2] );
	}
	free( resultCoords );

	/* 細分後の要素 */
	printf("Element : Count = %"PRIsz"¥n", refineElementCount );
	for(i=0;(size_t)i<refineElementCount;++i){
		printf("%d : (%d) %d, %d, %d, %d, %d, %d, %d, %d¥n", i+elementOffset, etype,
			refineHexas[8*i], refineHexas[8*i+1], refineHexas[8*i+2], refineHexas[8*i+3],
			refineHexas[8*i+4], refineHexas[8*i+5], refineHexas[8*i+6], refineHexas[8*i+7] );
	}

	/* 細分後の節点グループの更新 */
	ng0Count = rcapGetNodeGroupCount("ng0");
```

```
        result_ng0 = (int32_t*)calloc( ng0Count, sizeof(int32_t) );
        printf("Refined Node Group : Count = %"PRIsz"¥n", ng0Count );
        rcapGetNodeGroup("ng0",ng0Count,result_ng0);
        for(i=0;(size_t)i<ng0Count;++i){
                printf("%d¥n", result_ng0[i]);
        }
        free( result_ng0 );

        free( refineHexas );
        rcapTermRefiner();
        return 0;
}

#endif
```

1.6　形状関数による形状補正を行う

```
/*
 *
 * 形状関数による四面体 2 次要素の細分テスト用プログラム
 *
 * Sample Program for refinement by shape functions.
 * This model consists of one element of second degree tetrahedron.
 *
 */

#ifdef _CONSOLE

#include "rcapRefiner.h"
#include "rcapRefinerMacros.h"
#include <stdio.h>
#include <stdlib.h>    /* for calloc, free */
#include <assert.h>

int main(void)
{
        int32_t nodeOffset = 0;
```

```
int32_t elementOffset = 0;
/* 初期節点 */
size_t nodeCount = 10;
float64_t coords[30] = {
        0.0, 0.0, 0.0, // 0
        1.0, 0.0, 0.0, // 1
        0.0, 1.0, 0.0, // 2
        0.0, 0.0, 1.0, // 3
        0.5, 0.5,-0.1, // 4
        0.0, 0.5,-0.1, // 5
        0.5, 0.0,-0.1, // 6
        0.0, 0.0, 0.5, // 7
        0.5, 0.0, 0.5, // 8
        0.0, 0.5, 0.5   // 9
};
/* 細分後の節点 */
size_t refineNodeCount = 0;
float64_t* resultCoords = NULL;
/* 細分前の要素 */
int8_t etype = RCAP_TETRAHEDRON2;
size_t elementCount = 1;
int32_t tetras[10] = {
        0, 1, 2, 3, 4, 5, 6, 7, 8, 9
};
/* 細分後の要素 */
size_t refineElementCount = 0;
int32_t* refineTetras = NULL;

/* カウンタ */
int32_t i,j;

/* 初期化：節点番号、要素番号のオフセット値を与える */
rcapInitRefiner( nodeOffset, elementOffset );

/* 形状関数による中間節点を有効にする */
rcapSetSecondFitting(1);
```

```
	printf("----- Original Model -----¥n");
	/* 座標値を Refiner に与える */
	rcapSetNode64( nodeCount, coords, NULL, NULL );
	/* 細分前の節点数 */
	nodeCount = rcapGetNodeCount();
	assert( nodeCount == 10 );
	printf("Node：Count = %"PRIsz"¥n", nodeCount );
	for(i=0;(size_t)i<nodeCount;++i){
		printf("%d：%f, %f, %f¥n", i+nodeOffset, coords[3*i], coords[3*i+1],
coords[3*i+2] );
	}
	/* 細分前の要素数 */
	assert( elementCount == 1 );
	printf("Element：Count = %"PRIsz"¥n", elementCount );
	for(i=0;(size_t)i<elementCount;++i){
		printf("%d：(%d) %d, %d, %d, %d, %d, %d, %d, %d, %d, %d¥n",
i+elementOffset, etype,
			tetras[10*i], tetras[10*i+1], tetras[10*i+2],
tetras[10*i+3], tetras[10*i+4],
			tetras[10*i+5], tetras[10*i+6], tetras[10*i+7],
tetras[10*i+8], tetras[10*i+9]
			);
	}

	printf("----- Refined Model -----¥n");

	/* 要素の細分 */
	refineElementCount = rcapRefineElement( elementCount, etype, tetras,
NULL );
	refineTetras = (int32_t*)calloc( 10*refineElementCount, sizeof(int32_t) );
	elementCount = rcapRefineElement( elementCount, etype, tetras,
refineTetras);
	rcapCommit();

	/* 細分後の節点 */
```

```
        refineNodeCount = rcapGetNodeCount();
        printf("Node：Count = %"PRIsz"¥n", refineNodeCount );
        resultCoords = (float64_t*)calloc( 3*refineNodeCount, sizeof(float64_t) );
        rcapGetNodeSeq64( refineNodeCount, nodeOffset, resultCoords );
        for(j=0;(size_t)j<refineNodeCount;++j){
                printf("%d : %f, %f, %f¥n", j+nodeOffset, resultCoords[3*j],
resultCoords[3*j+1], resultCoords[3*j+2] );
        }
        free( resultCoords );

        /* 細分後の要素 */
        printf("Element：Count = %"PRIsz"¥n", refineElementCount );
        for(i=0;(size_t)i<refineElementCount;++i){
                printf("%d：(%d) %d, %d, %d, %d, %d, %d, %d, %d, %d, %d¥n",
i+elementOffset, etype,
                        refineTetras[10*i], refineTetras[10*i+1],
refineTetras[10*i+2], refineTetras[10*i+3], refineTetras[10*i+4],
                        refineTetras[10*i+5], refineTetras[10*i+6],
refineTetras[10*i+7], refineTetras[10*i+8], refineTetras[10*i+9]
                        );
        }
        free( refineTetras );

        rcapTermRefiner();
        return 0;
}

#endif
```

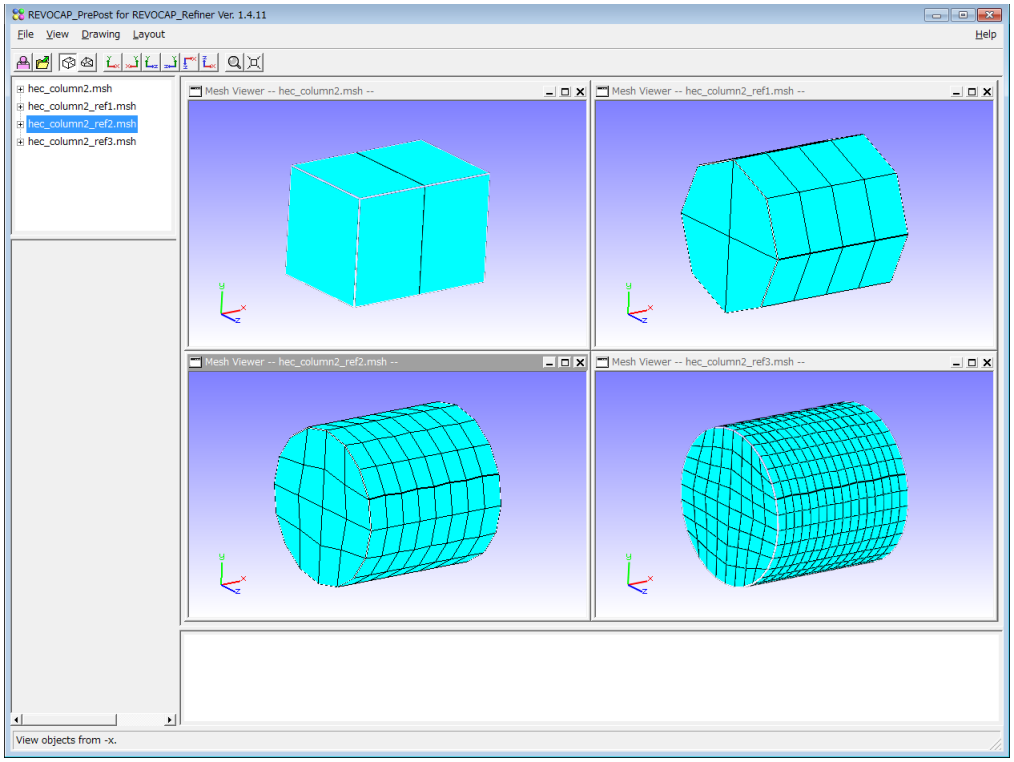
2 サンプルデータ

2.1 円柱

データは Refiner/data/column にある。

- hec_column.msh HEC_MW 形式の初期メッシュ

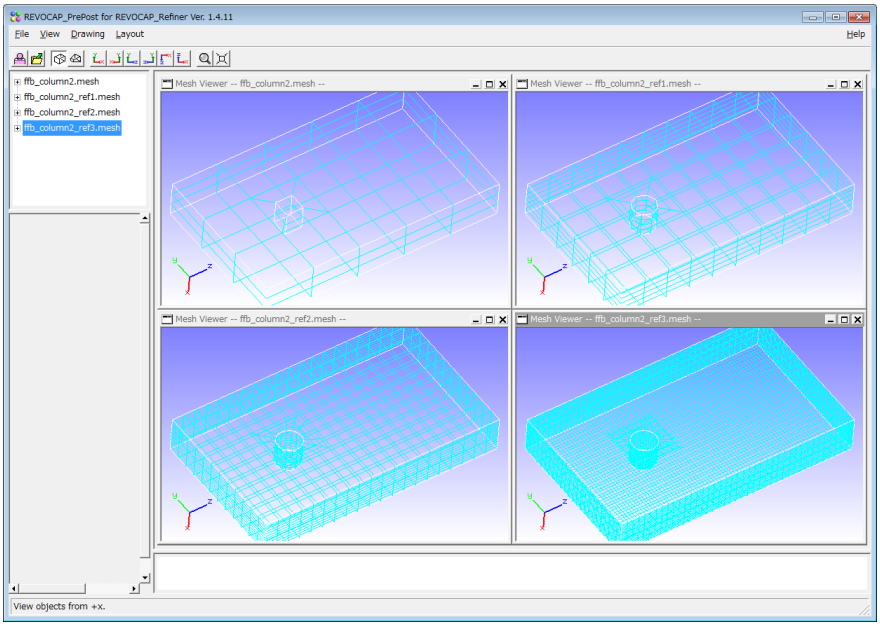
- ffb_column.mesh FrontFlow/blue GF 形式の初期メッシュ
- column.rnf 形状と初期適合データ
  領域分割後の局所節点番号のテストを兼ねて、初期適合データの節点番号は初期メッシュの節点番号から 100 ずれていることに注意する。
  形状は HEC_MW 形式、FrontFlow/blue GF 形式とも同一のものである。
  形状適合細分例は以下の通り。



2.2 円柱２

データは Refiner/data/column2 にある。

- hec_column2.msh HEC_MW 形式の初期メッシュ
- ffb_column2.mesh FrontFlow/blue GF 形式の初期メッシュ
- column2.rnf 形状と初期適合データ
  構造用（HEC_MW 形式）は円柱と同じで、流体用（FrontFlow/blue GF 形式）の初期メッシュを円柱の外部に取ったものである。
  形状適合細分例は以下の通り。



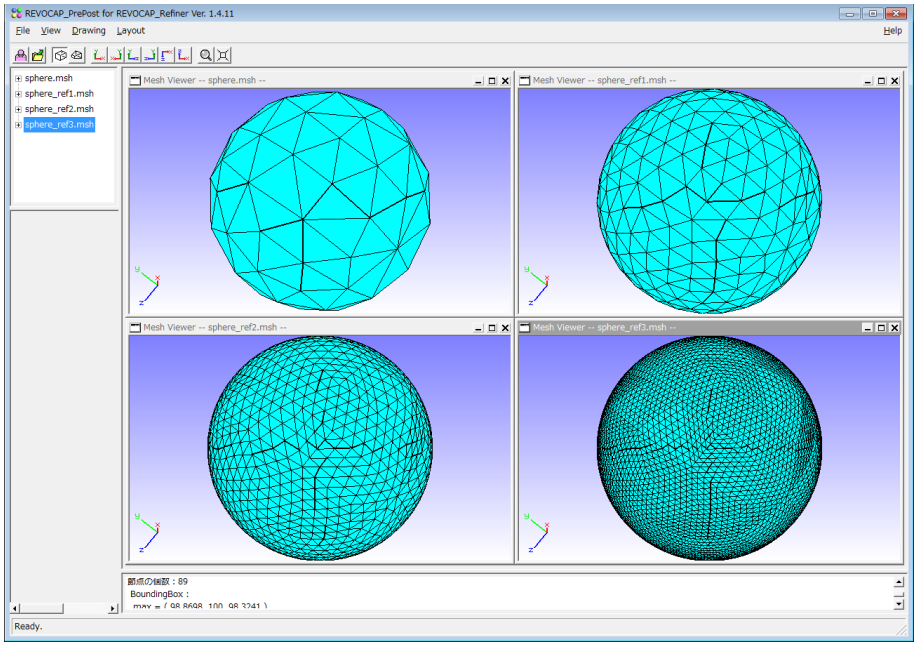
2.3 球面

データは Refiner/data/sphere にある。

- sphere.igs CAD 形状データ
- sphere.msh ADVENTURE_TetMesh 形式初期メッシュ
- sphere_fitting.rnf 形状と初期適合データ

  sphere.msh は ADVENTURE_TetMesh で生成されたメッシュであり、sphere_fitting.rnf は REVOCAP_PrePost で生成された初期適合データである。初期メッシュは形状適合の効果をみるためにあえて粗めに生成している（基準長さ=半径/2=50.0）。

  形状適合細分例は以下の通り。



2.4　ボトル形状

データは Refiner/data/bottle にある。

- bottle.mesh FrontFlow/blue GF 形式の初期メッシュ
- bottle_fitting.rnf 形状と初期適合データ

bottle.mesh は混合要素のテストのためのメッシュであり、四面体、六面体、三角柱、四角錐の各要素が混在している。

形状適合細分例は以下の通り。