ＩＳＳＮ 1341－6839

情報処理センター
研　究　報　告

The Bulletin of the Information Processing Center

第　31　号

（２０１０．３）

岡 山 理 科 大 学

Okayama University of Science

岡山市北区理大町１－１

Tel（086）252−3012（直）

目　　次

GPU を利用した計算効率の改善について

河野　敏行[*1], 吉岡　良将[*2], 新田　敏弘[*3]
[*1] 岡山理科大学総合情報学部情報科学科, [*2] 岡山理科大学総合情報学部研究員
[*3] 岡山理科大学大学院総合情報研究科博士課程 D2

1.　はじめに

近年, 多くの擬似3D ゲームの普及により, ビデオカードの性能が飛躍的に伸びたと考えられる. ビデオカードにおける演算処理装置は GPU(Graphics Processing Unit)と呼ばれる. これは画像処理にかかる CPU への負担を軽減するため, 画像処理に特化したさまざまな機能を持っている. そして多くの並列処理が可能となってきている. この GPU を画像処理以外の目的に応用する技術も発展し, GPGPU（General-purpose computing on graphics processing units）すなわち, GPU による汎用目的計算が普及している.

この論文においては, 情報処理センターに置かれた GPU 計算機による CUDA を用いたの計算機環境の確認と, その計算の実行方法を解説し, 各処理にかかる時間を比較し, 並列効率について調べた結果を示し, 考察をする.

2.　GPU 演算システムの概要

センターに設置された GPU 計算機システムには, 2 枚のビデオカードがある. Tesla　C1060 と Quadro　NVS290 である. Tesla　C1060 の概要を表 1 に示す.

表 1　Tesla　C1060

CUDA Driver Version	2.30
Total amount of global memory	4294705152 bytes （4GB）
Number of multiprocessors	30
Number of cores	240
Total amount of shared memory per block	16384 bytes（16KB）
Maximum sizes of each dimension of a block	512 x 512 x 64
Maximum sizes of each dimension of a grid	65535 x 65535 x 1
Clock rate	1.30 GHz

表 1 の環境情報は, CUDA のサンプルテンプレートの中のプログラムを図 1 のように実行することで得られる.

図 1　デバイス環境の確認コマンド

```
$ ~/NVIDIA_GPU_Computing_SDK/C/bin/linux/release/deviceQuery
```

その他の多くのサンプルが準備されており, 元のソースは

~/NVIDIA_GPU_Computing_SDK/C/src/

にある.

3　サンプルコードとプログラム開発環境

実験環境として，既存のコンパイル環境を利用することが容易なため，サンプルコードが置かれている ~/NVIDIA_GPU_Computing_SDK/C/src/ 以下にフォルダを作成し，Makefile により，make することとする．次に，簡単なプログラムを作成し，その作業と処理結果を示す．

3.1　スレッド番号を確認するプログラム

演算は設定により並列レベルを設定することが出来る．単純に，各スレッドの ID を表示するプログラムを示す．次に一連の作業を示す．

①フォルダ kakunin を作成する．

```
[kohno@gpu-comp src]$ mkdir kakunin
[kohno@gpu-comp src]$ cd kakunin
```

②CPU 側のプログラム main.cu を作成する．プログラムをプログラム 1 に示す．

プログラム 1 main.cu

```
#include <stdio.h>
#include <cutil.h>

#include "main_kernel.cu"

const int L=512;
const int M=1;
const int N=64;

int main( int argc, char** argv)
{
    // デバイスの初期化
    CUT_DEVICE_INIT(argc, argv);
    // 出力ファイル
    FILE *fp=fopen("result.txt","w");
    // タイマーの設定　開始
    unsigned int timer = 0;
    CUT_SAFE_CALL( cutCreateTimer( &timer));
    CUT_SAFE_CALL( cutStartTimer( timer));

    // ホストのデータメモリ確保　①
    int* h_joho = (int*) malloc(sizeof( int) *L*M*N);

    // デバイス上のデータメモリ確保　②
    int* d_joho;
    CUDA_SAFE_CALL( cudaMalloc( (void**) &d_joho, sizeof( int) * L*M*N));

    // デバイスへのデータのコピー　③
    CUDA_SAFE_CALL( cudaMemcpy( d_joho, h_joho, sizeof( int) *L*M*N,
cudaMemcpyHostToDevice));

    dim3 grid(N,1,1);
    dim3 threads(L,M,1);

    // GPU への計算　④
    mainkernel<<< grid, threads >>>(d_joho,L,M,N);

    // データのコピー　⑤
    CUDA_SAFE_CALL( cudaMemcpy(h_joho, d_joho, sizeof( int) *L*M*N,
cudaMemcpyDeviceToHost));
```

```
    CUT_SAFE_CALL( cutStopTimer( timer));
    printf( "Processing time: %f (ms)¥n", cutGetTimerValue( timer));
    CUT_SAFE_CALL( cutDeleteTimer( timer));

    for(int i=0;i<L*M*N;i++)
      fprintf(fp,"%d¥t%d¥n",i,h_joho[i]);

    free( h_joho);
    CUDA_SAFE_CALL(cudaFree(d_joho));
    fclose(fp);

    CUT_EXIT(argc, argv);
    return 0;
}
```

③GPU 側のプログラム(カーネル)　main_kernel.cu を作成する.

プログラム 2　main_kernel.cu

```
__global__ void mainkernel(int* g_joho,int L, int M, int N)
{
    const unsigned int tidx =threadIdx.x;
    const unsigned int tidy =threadIdx.y;
    const unsigned int bidx =blockIdx.x;

    int id = bidx*L*M+tidx*M+tidy;

    g_joho[id]=id;
}
```

④Makefile を設定する.

ここでは, コンパイル情報を記述しておく.

プログラム3　Makefile

```
# Add source files here
EXECUTABLE          := kakunin.exe
# Cuda source files (compiled with cudacc)
CUFILES             := main.cu
# C/C++ source files (compiled with gcc / c++)
CCFILES             := ¥

include ../../common/common.mk
```

⑤make　する.

```
[kohno@gpu-comp kakunin]$ make
```

⑥プログラムを実行する.

　実行ファイルが指定されたディレクトリに作成される.

```
$ ~/NVIDIA_GPU_Computing_SDK/C/bin/linux/release/kakunin.exe
Using device 0: Tesla C1060
Processing time: 89.751999 (ms)

Press ENTER to exit...
```

以上の結果は，L=512，M=1，N=64 のデータ数 32768 に対する結果である．このプログラムを利用して，各処理にかかる時間について確認を行う．表 2 から分かるように，ビデオカード上にメモリを確保する時間がほとんどである．

表 2　GPU のスレッド数の違いと各処理時間

L	M	N	size	①	②	③	④	⑤	total time(ms)
512	1	64	32768	0.007	89.857	0.090	0.036	0.152	90.142
16	16	16	4096	0.004	90.184	0.014	0.025	0.037	90.264
128	4	16	8192	0.004	90.858	0.028	0.035	0.058	90.983
1	512	64	32768	0.007	86.153	0.072	0.026	0.120	86.378

各 GPU に処理されるデータ量は微小であるため，計算時間自体はわずか(④)であり，デバイス上のメモリ確保(②)が全体の計算時間を占めていることがわかる．計算時間は実行ごとにばらつきがあり，この実験では単にメモリの確保に時間があることを確認した．スレッドを 1 次元とした場合，多少メモリ確保時間が少ないことが分かった．

3.2　1GPU による処理

メインプログラムからカーネルを呼び出す際に，1 スレッドのみを利用する．

```
mainkernel<<< 1,1 >>>(d_joho,L,M,N);
```

カーネルのプログラムは次のように書き直す．

```
__global__ void mainkernel(int* g_joho,int L, int M, int N)
{
    for(int i=0;i<N; i++){
      for(int j=0;j<M; j++){
          for(int k=0;k<L; k++){
              int id = i*L*M+j*M+k;
              g_joho[id]=id;
          }
      }
    }
}
```

1GPU で実行した結果，L=512，M=1，N=64 に対する演算時間は 92.04ms 程度であった．

3.3　1CPU による処理

次のプログラムを実行した結果，0.10ms 程度であった．実際の計算部分④と比較すると多少であるが，並列処理が速いことが分かった．主要処理は高速であるが，全体の処理ではメモリの確保が大きいので考慮する必要がある．

```
    // CPU での計算
   for(int i=0;i<N; i++){
      for(int j=0;j<M; j++){
          for(int k=0;k<L; k++){
              int id = i*L*M+j*M+k;
              h_joho[id]=id;
          }
      }
   }
```

次に行列の掛け算の比較をメモリの確保を含めたトータル時間で比較を行う．

4．行列の掛け算の比較

N 次の正方行列 A と B を乗算する計算を行う．GPU に対して，スレッドとブロックを用意して計算を行う．行列 A と B の要素は float で任意な値を与える．

表 3　行列の掛け算

N	スレッド数	ブロック数	Total　time（ms）
16	4×4	4×4	89.647003
16	16×16	1	87.177750
256	16×16	16×16	90.477200
256	256×256	1	89.360199

N=256 の場合，GPU にて 1 スレッドで計算すると 8196ms かかった．このとき並列化によって，10 倍ほどの高速化が図れることが確認できた．ちなみに CPU で実行した場合，44.055ms であった．上記の表 3 では，メモリの確保とメモリへの転送時間が含まれており，1 回の乗算に対して，全てのデータのコピー，すなわち，Host と Device のメモリコピーの無駄があり，表 2 に示した通りである．実際に数値計算においては，メモリのやり取りは最小にする必要がある．そのことを踏まえて，次の実験を行う．

5．2 次元偏微分方程式の差分法

2 次元のラプラス方程式を等間隔の格子が張られた正方領域について中心差分を用いて計算を行う．解法としては，基本反復法を用いることとする．条件として，図1に示すようにディリクレ境界条件を与え，1辺を N+1 等分し，内分点 N×N の未知数を計算する．初期値は 0.0 とする．反復解法として，point-Jacobi 法を用い，1 スレッド当たり 1 未知数を用いるように並列化を行う．

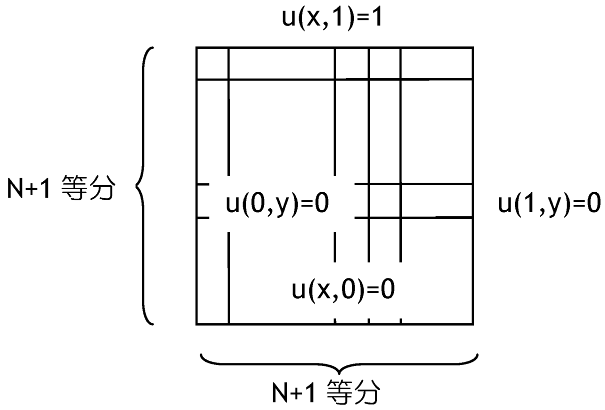

図1　シミュレーション条件

反復回数を 1000 回と決めて，毎回，メモリをコピーした場合と反復計算後にデータをコピーした場合の計算時間を比較する．Jacobi 法は各要素の計算が独立しており，各ビデオカードでの計算を繰り返すだけで済むため，ホストへのメモリをコピーする必要無く，反復計算を行うことが可能である．すなわち，表 4 に示すように毎回のメモリコピーなしに計算を行うことで，演算時間が大幅に縮小されることが分かる．CPU で計算をする場合，逐次処理を行うため1回の反復計算に N の 2 乗のループが必要であり，さらに Jacobi 法では，k 回目の演算で得られた各要素の最新データは k 回目では使わないため，全要素の計算終了後に更新を行う必要があり，そのため N の2乗のループが必要になる．したがって，次数が増加することで多くの演算時間が必要となっている．

表 4　point-Jacobi 法の計算時間（ms）

N	スレッド数	ブロック数	毎回メモリコピー (ite.+memcpy)	毎回のメモリコピーなし (ite. only)	CPU
16	16×16	1×1	123.146(33.726)	98.464(1.4334)	1.095
256	16×16	16×16	474.961(387.426)	254.835(1.4704)	851.0158
512	16×16	32×32	1728.582(1641.187)	947.585(1.4332)	2862.2846
1024	16×16	64×64	5440.131(5354.615)	3438.058(2.2950)	7713.8316

この実験では，反復回数を 1000 回と決めてトータル時間を計測した．N＝16 のとき 256 の未知数に対して，N=1024 のとき，1048576 と 4096 倍の未知数の計算している．各スレッドの計算量は変わっていないが，実計算時間は 1.4334ms から 2.2950ms の約 2 倍となっている．先の実験からも明らかであるが，毎回，デバイス（GPU）で計算された値をホスト（CPU）にコピ

ーした場合は，反復計算回数分の処理時間がかかり，効率的ではないことが明らかである．しかしながら，実際の数値計算を行う際は，収束状況を調べるために，収束判定を行う必要がある．すなわち，計算された値を受け取り，CPU 側でノルムの計算を行うなどのアルゴリズムが必要となり，N=1024 では，5440.131ms と同様だけ時間がかかると思われる．

1CPU で計算した結果と比較すると，N=16 すなわち，256 個の未知数に対する演算は GPU 側のメモリの確保を考えると CPU で処理した方が早かったが，N=256 以上すなわち，65536 個以上の未知数の計算では，GPU にて並列計算した方がメモリの確保，そしてメモリの転送を含めて速いことが分かった．

5.1 収束判定を含めた計算結果

収束判定条件として，k 回目の反復によって得られた近似解$\mathbf{x}^{(k)}$に対して，下記の式を利用する．

$$\left\| x^{(k+1)} - x^{(k)} \right\| < 1.0 \times 10^{-5}$$

N=256 に対して 16×16 ブロックで 1 ブロックに 16×16 スレッドによって各値を 1 スレッドで解く．試す方法として，反復回ごとに収束チェックして行う方法と反復 10 回ごとに memcpy でデータを呼び出し，収束判定を行う方法，そして，GPU 内で反復を 10 回行う方法を実験した．数値結果は表 5 に示す．

表 5 収束判定を行った結果

方法	反復回数	処理時間(ms)
CPU で計算	81624	23348.072
反復回ごとに収束チェック	81119	42628.848
反復 10 回ごとに収束チェック	81191	17234.422
GPU 内で 10 回反復	62040	22727.617

計算方法が Jacobi 法であるため，反復回数は 8 万回程度かかっているが，GPU 内で 10 回反復する場合は，その反復内でデータの更新が一部行われている可能性があり，反復回数は減っている．すなわち，不規則な Gauss-Seidel 法のような反復が行われ，反復回数が減ったと考える．1CPU で計算した場合，23 秒程度かかっているが，反復 10 回ごとに収束チェックした場合は，17 秒程度と効率が良くなっている．チェックの方法はさらに改良が必要であると考える．

6. まとめ

CUDA を利用した数値計算では，CPU と GPU でのメモリの扱い方，そして，並列処理のためのスレッドとブロックそして，カーネルを複数利用する場合はグリッドについて理解して使う必要があることが分かった．今回は，point-Jacobi 法をスレッド数に合わせた次数で解くことを

考えた．その結果，通常の CPU 演算より GPU による並列処理が高速であることが示された．反復ごとのメモリ転送は，毎回必要ではなく，数回ごとに行うことの工夫が考えられ，高速にする方法はいくつか考えられる．

今後の課題として，問題をさらに大規模にした場合は，スレッド数の限界があるため，扱う問題をブロック行列とするなどの工夫が考えられる．しかしながら，これまでの環境と比べると，GPU は並列化が容易でありプログラムしやすい．そして，今回使用した C1060 では 512×512×64 のブロック化ができ，大規模な並列環境を構築できる．さらに，大規模な処理の実験は今後の課題である．

今回の実験では，メモリの確保とメモリの転送が処理時間に影響することを述べたが，CUDA では，グローバルメモリだけではなく，シェアードメモリを利用することもできる．シェアードメモリは同一ブロック内のスレッドからしかアクセスできないが，グローバルメモリに比べて高速にアクセスできる．そして，同様に高速なコンスタントメモリはカーネル内から変更はできないが，CPU 側から更新することができ，同様に全体の高速化が可能と思われる．これらのメモリの使い方の違いについても実験することが残されている．さらに，GPU が処理されている間，CPU は空いているため，CPU＋GPU の並列処理を行うことで，高速化が図れると思われる．そして，センターの計算機にはもう一枚のビデオカード NVS2940 も搭載されており，2 つの GPU 装置を使っての計算についても今後の課題である．

参考文献

[1] 株式会社 爆発研究所，High Performance Computing Quick Start Guide for Multi GPU BAKUSOKU，2009.

[2] 小国力訳，コンピュータによる連立一次方程式の解法，丸善株式会社，1993.

[3] CUDA ZONE，http://forum.nvidia.co.jp/EokpControl?&event=TE0001.

教育用分子軌道計算システム eduDV の開発

岡山理科大学 理学部 化学科 坂根弦太

gsakane@chem.ous.ac.jp

1. はじめに

化学の土台となる電子の本当の姿を理解するには，量子力学に基づく原子や分子の電子状態を理解しなければならない．これを取り扱う量子化学は，理系の大学に入った学生にとって基礎化学の中でも際立って学習が困難な項目である．学生は高等学校で習った電子の惑星モデル（長岡半太郎やラザフォードの原子模型）の概念（図１）を捨て，シュレディンガー方程式を解いて得られる原子軌道関数について学ぶ．しかし初めて目にする記号や方程式に途惑い，その具体的なイメージを抱けずに化学に対する興味を失ってしまう学生も少なくない．

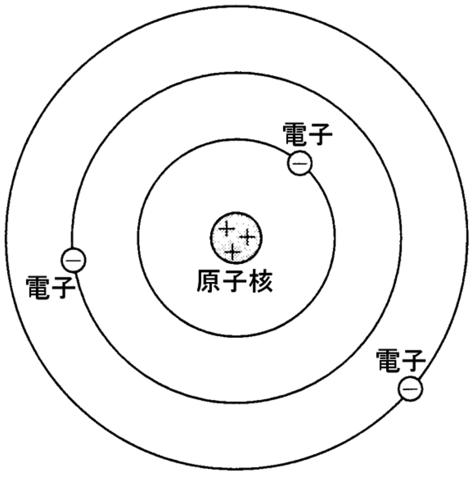

図１　高等学校で学ぶ電子のイメージ

大学教員にとっても，主量子数，方位量子数，磁気量子数で決まる周期表の各元素の原子軌道，およびその原子軌道を線形結合して得られる分子軌道の三次元的なイメージおよびそれらのエネルギー準位について，限られた講義時間内で学生に正確に把握させることは，教科書や黒板への板書だけでは極めて困難である．例えば 2p$_x$ 軌道を例にとると，教科書[1]の図（図２）は，本稿で紹介するシステムで描いた図（図３）とはかなり違っている．

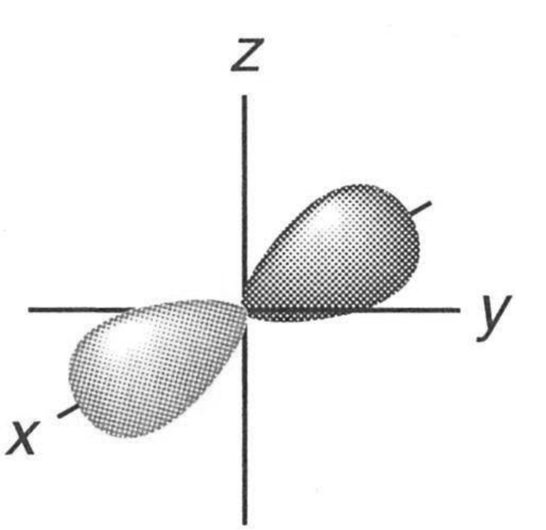

図２　大学の教科書に掲載されている波動関数

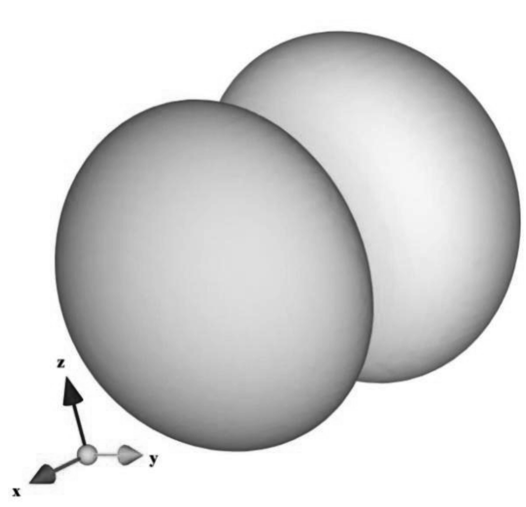

図３　本稿のシステムで計算・表示した波動関数

物性物理や材料科学分野研究の第一線で使われている分子軌道計算プログラムにDV−Xα法がある[2]．周期表全元素を同じ精度で取り扱うことができ，比較的大きな分子であっても普通のパソコンで短時間かつ精度よく電子状態を計算できる[3]．最新の実行ファイルはウェブからダウンロードし，教育研究用には無償で使える[4]．さらにシェアウェアである秀丸エディタ[5]をインストールし（アカデミックフリー制度があり，学校内に設置しているパソコンで学生が利用する場合は，申請により適用される），泉富士夫氏が開発した秀丸エディタマクロ集“DV−Xα法計算支援環境”[6]を組み込むことにより GUI でDV−Xα法が利用できるようになる[7]．泉富士夫氏，門馬綱一氏が開発した“結晶構造，及び電子・核密度等の三次元可視化プログラム VESTA”[8]をダウンロード[9]・インストールすれば，DV−Xα法で計算した波動関数，電子密度，静電ポテンシャルなどを三次元可視化できるようになる[7]．

量子化学計算に熟練した研究者が使用するには以上の環境で十分だが，大学１・２年次生に講義時間内でこれを操作させるのは以下に挙げる理由により困難である．

・CP/M−86 や MS−DOS など，Character-based User Interface(CUI)の OS でパソコンを使っていた世代とは異なり，現在の学生は，最初に触れたパソコンからすでに Windows など Graphical User Interface(GUI)を実装した OS であった世代である．インターネットのWWWブラウザ，ワープロ・表計算ソフト，プレゼンテーションソフトなどは高等学校「情報」の授業および大学での「パソコン入門」系の講義で簡単な操作法は修得しているものの，化学系の学生では unix を扱うこともなく，プログラミングの経験もほとんどない．CUI でコンピュータを操作したという経験がほとんど無いため，簡単な MS−DOS コマンドはもとより，ファイル（テキストファイル，バイナリファイル）やディレクトリ（フォルダ）のツリー構造などの概念が構築されていない学生が多い．Fortran77 も学んだことがないためファイルの入出力，実数型や整数型の書式，ファイルの種類（テキスト，バイナリ），装置番号というような概念がない．しかし DV−Xα分子軌道計算プログラムの計算環境は，Windows のコマンドプロンプトで作業していく CUI をベースとしたシステムであり，計算作業ディレクトリを作成したりファイルをコピーしたりするなどの必要最低限の MS−DOS コマンドは修得している必要がある．また DV−Xα分子軌道計算プログラム本体は Fortran77 で書かれているため，その入力ファイルをエディタ等で準備するにあたっては，必要最低限の Fortran の書式の概念（実数型・整数型・指数型・文字型変数記述子など）も必要である．これらを授業の中で最初から説明していると，それだけで数時間を要し，情報リテラシーの授業になってしまう（それはそれで意味のある教育ではあるが）．

・DV−Xα分子軌道計算プログラムの入力ファイル F01 を作成するのに，直線型の窒素分子，一酸化炭素分子，二酸化炭素分子程度ならまだしも，水分子やアンモニア分子やメタン分子となってくると，その座標を求めるだけでかなりの時間を要する．例えば三角関数が分かっていない学生が多い場合，まるで数学の幾何の授業のようになってしまう（それはそれで重要な教育ではあるが）．パソコン画面上で三次元的に分子の棒球モデルを表示することにより分子を構築して座標を自動出力するなら問題ないであろうが，紙と鉛筆だけで三次元的な分子の座標を自分で算出することができない学生は多い．

・無機化学や量子化学などの教科書に掲載されている小分子や単核錯体の分子軌道エネルギー準位図には，その分子・錯体の対称性（属する点群）に基づく分子軌道の名称がついている．学生に点群の指標

表の重要性を実感させるためにも，縮退するはずの分子軌道はきちんと縮退させるためにも，正しい対称軌道(F25)を入れて DV−Xα分子軌道計算を行いたいが，DV−Xα用の対称軌道作成プログラム symOrb（「はじめての電子状態計算[3]」添付 CD−ROM に同梱)を動かすためには Mathematica が必要である．しかし Mathematica は高価であり，学校現場のパソコンにはインストールされていないことが多い（化学系研究室でも持っていないことが多い)．

そこで筆者は，限られた講義時間内で高校生でも大学生でも，何の予備知識もなしに，いきなり周期表元素の原子軌道や，教科書に掲載されている様々な分子の分子軌道を自ら計算して，電子（波動関数）の形状やそのエネルギーを眺めることができる“教育用分子軌道計算システム eduDV”[10]の開発に着手した．今年度はバグフィックス，コンパイラーのバージョンアップに伴う再コンパイル作業を行うと共にマニュアル[11]の整備，論文発表[12]による啓蒙・普及活動に努めた．

マニュアル[11]に従ってダウンロード[13]・インストールすれば環境は整う．岡山理科大学情報処理センターの学生実習用の全パソコンには DV−Xα法[4]，秀丸エディタ[5]，DV−Xα法計算支援環境[6]，VESTA[9]，eduDV[13]がインストールされており，筆者の担当する講義・実習で大いに活用している．

2. 開発環境

情報処理センターより貸与されたノートパソコン（東芝 Dynabook SS 2010 DS86P/2, CPU: Mobile Intel Pentium III 866 MHz, RAM: 256 MB, OS: Windows 2000 Service Pack 4）に Open Watcom Fortran77 compiler (Version 1.9)[14]をインストールした環境を用いた．

3. プログラム構成

教育用分子軌道計算システム eduDV は，以下のプログラム群から構成されており，筆者のウェブサイト[13]よりダウンロードできる．

3-1. Fortran 77 ソースプログラム

01. atom.f	**143,202 B**	**21. cnvchk200.f**	**2,550 B**
02. atomn.f	**143,378 B**	**22. cnvchk250.f**	**2,550 B**
03. c2v12.f	**4,253 B**	**23. cnvchk300.f**	**2,550 B**
04. c2v12n.f	**4,369 B**	**24. cnvchk50.f**	**2,549 B**
05. c2v12s.f	**4,365 B**	**25. cscafix.f**	**2,693 B**
06. c3edfix.f	**1,987 B**	**26. d3h13.f**	**3,728 B**
07. c3v13.f	**4,605 B**	**27. d3h13n.f**	**3,812 B**
08. c3v13n.f	**4,721 B**	**28. d3h13s.f**	**3,736 B**
09. c3v13s.f	**4,717 B**	**29. d4h14.f**	**4,278 B**
10. c8v11.f	**3,395 B**	**30. d4h14n.f**	**4,291 B**
11. c8v111.f	**4,290 B**	**31. d4h14s.f**	**4,287 B**
12. c8v1111.f	**4,894 B**	**32. d6h66.f**	**5,491 B**
13. c8v1111n.f	**4,972 B**	**33. d6h66n.f**	**5,504 B**
14. c8v1111s.f	**4,968 B**	**34. d6h66s.f**	**5,500 B**
15. c8v111n.f	**4,369 B**	**35. d8h12.f**	**3,628 B**
16. c8v111s.f	**4,365 B**	**36. d8h12n.f**	**3,709 B**
17. c8v11n.f	**3,476 B**	**37. d8h12s.f**	**3,705 B**
18. c8v11s.f	**3,472 B**	**38. d8h2.f**	**3,212 B**
19. cnvchk100.f	**2,550 B**	**39. d8h22.f**	**4,004 B**
20. cnvchk150.f	**2,550 B**	**40. d8h22n.f**	**4,084 B**

41. d8h22s.f	4,080 B	63. ml6s.f	4,615 B
42. d8h2n.f	3,239 B	64. molda2dv.f	6,203 B
43. d8h2s.f	3,235 B	65. oh16.f	4,174 B
44. dv2chem3d.f	3,510 B	66. oh166.f	5,454 B
45. dv2daph.f	10,436 B	67. oh166n.f	5,571 B
46. dv2f01.f	7,414 B	68. oh166s.f	5,567 B
47. dv2molda.f	6,724 B	69. oh16n.f	4,239 B
48. endsts.f	3,264 B	70. oh16s.f	4,235 B
49. existf05.f	1,034 B	71. p3edadd.f	2,076 B
50. f01add.f	2,587 B	72. p3edfix.f	2,210 B
51. hlgap.f	2,365 B	73. prests.f	2,874 B
52. hlgaps.f	3,802 B	74. prests2.f	1,336 B
53. ion.f	145,533 B	75. prestsL.f	2,876 B
54. ionn.f	145,534 B	76. pscaadd.f	2,544 B
55. mh2o6.f	6,016 B	77. pscafix.f	2,957 B
56. mh2o6n.f	6,141 B	78. read36.f	13,325 B
57. mh2o6s.f	6,137 B	79. td14.f	3,920 B
58. ml4.f	4,357 B	80. td144.f	4,907 B
59. ml4n.f	4,369 B	81. td144n.f	5,024 B
60. ml4s.f	4,365 B	82. td144s.f	5,020 B
61. ml6.f	4,607 B	83. td14n.f	3,985 B
62. ml6n.f	4,619 B	84. td14s.f	3,981 B

3-2. Open Watcom Fortran 77 Version 1.9 等でコンパイルした実行ファイル

01. atom.exe	34. d6h66s.exe	67. mh2o6.exe
02. atomn.exe	35. d8h12.exe	68. mh2o6n.exe
03. c2v12.exe	36. d8h12n.exe	69. mh2o6s.exe
04. c2v12n.exe	37. d8h12s.exe	70. ml4.exe
05. c2v12s.exe	38. d8h2.exe	71. ml4n.exe
06. c3edfix.exe	39. d8h22.exe	72. ml4s.exe
07. c3v13.exe	40. d8h22n.exe	73. ml6.exe
08. c3v13n.exe	41. d8h22s.exe	74. ml6n.exe
09. c3v13s.exe	42. d8h2n.exe	75. ml6s.exe
10. c8v11.exe	43. d8h2s.exe	76. molda2dv.exe
11. c8v111.exe	44. dv2chem3d.exe	77. ncontrd.exe
12. c8v1111.exe	45. dv2daph.exe	78. ncontrdall.exe
13. c8v1111n.exe	46. dv2f01.exe	79. nmakec04d.exe
14. c8v1111s.exe	47. dv2molda.exe	80. oh16.exe
15. c8v111n.exe	48. endsts.exe	81. oh166.exe
16. c8v111s.exe	49. existf05.exe	82. oh166n.exe
17. c8v11n.exe	50. f01add.exe	83. oh166s.exe
18. c8v11s.exe	51. gcontrd.exe	84. oh16n.exe
19. cnvchk100.exe	52. gcontrdall.exe	85. oh16s.exe
20. cnvchk150.exe	53. gmakec04d.exe	86. p3edadd.exe
21. cnvchk200.exe	54. hcontrd.exe	87. p3edfix.exe
22. cnvchk250.exe	55. hcontrdall.exe	88. prests.exe
23. cnvchk300.exe	56. hlgap.exe	89. prests2.exe
24. cnvchk50.exe	57. hlgaps.exe	90. prestsL.exe
25. cscafix.exe	58. hmakec04d.exe	91. pscaadd.exe
26. d3h13.exe	59. icontrd.exe	92. pscafix.exe
27. d3h13n.exe	60. icontrdall.exe	93. read36.exe
28. d3h13s.exe	61. imakec04d.exe	94. scontrd.exe
29. d4h14.exe	62. ion.exe	95. scontrdall.exe
30. d4h14n.exe	63. ionn.exe	96. smakec04d.exe
31. d4h14s.exe	64. jcontrd.exe	97. td14.exe
32. d6h66.exe	65. jcontrdall.exe	98. td144.exe
33. d6h66n.exe	66. jmakec04d.exe	99. td144n.exe

100. td144s.exe
101.td14n.exe
102.td14s.exe
103.ucontrd.exe
104.ucontrdall.exe
105. umakec04d.exe
106. wcontrd.exe
107. wcontrdall.exe
108. wmakec04d.exe
109. xcontrd.exe
110. xcontrdall.exe
111. xmakec04d.exe
112. ycontrd.exe
113. ycontrdall.exe
114. ymakec04d.exe

※ gcontrd.exe, gcontrdall.exe, gmakec04d.exe, hcontrd.exe, hcontrdall.exe, hmakec04d.exe, icontrd.exe, icontrdall.exe, imakec04d.exe, jcontrd.exe, jcontrdall.exe, jmakec04d.exe, ncontrd.exe, ncontrdall.exe, nmakec04d.exe, scontrd.exe, scontrdall.exe, smakec04d.exe, ucontrd.exe, ucontrdall.exe, umakec04d.exe, wcontrd.exe, wcontrdall.exe, wmakec04d.exe, xcontrd.exe, xcontrdall.exe, xmakec04d.exe, ycontrd.exe, ycontrdall.exe, ymakec04d.exe は，大阪大学の水野正隆先生からご提供（使用許可）いただいたプログラムである．

3-3. Open Watcom Fortran 77 Version 1.9 でのコンパイル実行バッチファイル

01. latom.bat
02. latomn.bat
03. lc2v12.bat
04. lc2v12n.bat
05. lc2v12s.bat
96. lc3edfix.bat
07. lc3v13.bat
08. lc3v13n.bat
09. lc3v13s.bat
10. lc8v11.bat
11. lc8v111.bat
12. lc8v1111.bat
13. lc8v1111n.bat
14. lc8v1111s.bat
15. lc8v111n.bat
16. lc8v111s.bat
17. lc8v11n.bat
18. lc8v11s.bat
19. lcnvchk100.bat
20. lcnvchk150.bat
21. lcnvchk200.bat
22. lcnvchk250.bat
23. lcnvchk300.bat
24. lcnvchk50.bat
25. lcscafix.bat
26. ld3h13.bat
27. ld3h13n.bat
28. ld3h13s.bat
29. ld4h14.bat
30. ld4h14n.bat
31. ld4h14s.bat
32. ld6h66.bat
33. ld6h66n.bat
34. ld6h66s.bat
35. ld8h12.bat
36. ld8h12n.bat
37. ld8h12s.bat
38. ld8h2.bat
39. ld8h22.bat
40. ld8h22n.bat
41. ld8h22s.bat
42. ld8h2n.bat
43. ld8h2s.bat
44. ldv2chem3d.bat
45. ldv2daph.bat
46. ldv2f01.bat
47. ldv2molda.bat
48. lendsts.bat
49. lexistf05.bat
50. lf01add.bat
51. lhlgap.bat
52. lhlgaps.bat
53. lion.bat
54. lionn.bat
55. lmh2o6.bat
56. lmh2o6n.bat
57. lmh2o6s.bat
58. lml4.bat
59. lml4n.bat
60. lml4s.bat
61. lml6.bat
62. lml6n.bat
63. lml6s.bat
64. lmolda2dv.bat
65. loh16.bat
66. loh166.bat
67. loh166n.bat
68. loh166s.bat
69. loh16n.bat
70. loh16s.bat
71. lp3edadd.bat
72. lp3edfix.bat
73. lprests.bat
74. lprests2.bat
75. lprestsL.bat
76. lpscaadd.bat
77. lpscafix.bat
78. lread36.bat
79. ltd14.bat
80. ltd144.bat
81. ltd144n.bat
82. ltd144s.bat
83. ltd14n.bat
84. ltd14s.bat

3-4. プログラム実行バッチファイル

01. a2.bat
02. a2b2.bat
03. a2b2n.bat
04. a2b2s.bat
05. a2n.bat
06. a2s.bat
07. ab.bat
08. ab2.bat
09. ab2n.bat
10. ab2s.bat
11. abc.bat
12. abcd.bat
13. abcdn.bat
14. abcds.bat
15. abcn.bat
16. abcs.bat
17. abn.bat
18. abs.bat
19. atom.bat
20. atomn.bat
21. atoms.bat
22. c2v12.bat
23. c2v12n.bat
24. c2v12s.bat
25. c3edfix.bat
26. c3v13.bat
27. c3v13n.bat
28. c3v13s.bat
29. c8v11.bat
30. c8v111.bat
31. c8v1111.bat
32. c8v1111n.bat
33. c8v1111s.bat
34. c8v111n.bat
35. c8v111s.bat
36. c8v11n.bat
37. c8v11s.bat
38. cnvchk100.bat
39. cnvchk150.bat
40. cnvchk200.bat
41. cnvchk250.bat
42. cnvchk300.bat

43. cnvchk50.bat
44. contrd.bat
45. contrdall.bat
46. cscafix.bat
47. d3h13.bat
48. d3h13n.bat
49. d3h13s.bat
50. d4h14.bat
51. d4h14n.bat
52. d4h14s.bat
53. d6h66.bat
54. d6h66n.bat
55. d6h66s.bat
56. d8h12.bat
57. d8h12n.bat
58. d8h12s.bat
59. d8h2.bat
60. d8h22.bat
61. d8h22n.bat
62. d8h22s.bat
63. d8h2n.bat
64. d8h2s.bat
65. dv2chem3d.bat
66. dv2daph.bat
67. dv2f01.bat
68. dv2molda.bat
69. endsts.bat
70. existf05.bat
71. f01add.bat
72. hlgap.bat
73. hlgaps.bat
74. ion.bat
75. ionn.bat
76. makec04d.bat
77. makeEduDV.bat
78. mh2o6.bat
79. mh2o6n.bat
80. mh2o6s.bat
81. ml4.bat
82. ml4n.bat
83. ml4s.bat
84. ml6.bat
85. ml6n.bat
86. ml6s.bat
87. molda2dv.bat
88. oh16.bat
89. oh166.bat
90. oh166n.bat
91. oh166s.bat
92. oh16n.bat
93. oh16s.bat
94. p3edadd.bat
95. p3edfix.bat
96. prests.bat
97. prests2.bat
98. prestsL.bat
99. pscaadd.bat
100. pscafix.bat.
101. read36.bat.
102. td14.bat.
103. td144.bat.
104. td144n.bat.
105. td144s.bat.
106. td14n.bat.
107. td14s.bat.

※contrd.bat, contrdall.bat, makec04d.bat は，大阪大学の水野正隆先生からご提供（使用許可）いただいたプログラムである.

3-5. データファイル

01. 1.b05
02. 10.b05
03. 11.b05
04. 12.b05
05. 13.b05
06. 14.b05
07. 15.b05
08. 16.b05
09. 17.b05
10. 18.b05
11. 19.b05
12. 2.b05
13. 20.b05
14. 21.b05
15. 22.b05
16. 23.b05
17. 24.b05
18. 25.b05
19. 26.b05
20. 27.b05
21. 28.b05
22. 29.b05
23. 3.b05
24. 30.b05
25. 31.b05
26. 32.b05
27. 33.b05
28. 34.b05
29. 35.b05
30. 36.b05
31. 37.b05
32. 38.b05
33. 39.b05
34. 4.b05
35. 40.b05
36. 41.b05
37. 42.b05
38. 43.b05
39. 44.b05
40. 45.b05
41. 46.b05
42. 47.b05
43. 48.b05
44. 49.b05
45. 5.b05
46. 50.b05
47. 51.b05
48. 52.b05
49. 53.b05
50. 54.b05
51. 55.b05
52. 56.b05
53. 57.b05
54. 58.b05
55. 59.b05
56. 6.b05
57. 60.b05
58. 61.b05
59. 62.b05
60. 63.b05
61. 64.b05
62. 65.b05
63. 66.b05
64. 67.b05
65. 68.b05
66. 69.b05
67. 7.b05
68. 70.b05
69. 71.b05
70. 72.b05
71. 73.b05
72. 74.b05
73. 75.b05
74. 76.b05
75. 77.b05
76. 78.b05
77. 79.b05
78. 8.b05
79. 80.b05
80. 81.b05
81. 82.b05
82. 83.b05
83. 84.b05
84. 85.b05
85. 86.b05
86. 87.b05
87. 88.b05
88. 89.b05
89. 9.b05
90. 90.b05
91. 91.b05
92. 92.b05
93. 93.b05
94. 94.b05
95. atomf
96. c2v12
97. c3v13
98. c8v11
99. c8v111
100. c8v1111
101. d3h13
102. d4h14
103. d6h66
104. d8h12
105. d8h2
106. d8h22
107. mh2o6
108. oh16
109. oh166
110. one
111. td14
112. td144
113. three
114. two
115. zero

3-6. 秀丸エディタマクロ（秀丸エディタマクロ集“DV–Xα法計算支援環境”[6]に同梱）

01. eduDV.mac

02. a2.mac
03. a2b2.mac
04. a2b2menu.mac
05. a2b2n.mac
06. a2b2s.mac
07. a2menu.mac
08. a2n.mac
09. a2s.mac
10. ab.mac
11. ab2.mac
12. ab2menu.mac
13. ab2n.mac
14. ab2s.mac
15. abc.mac
16. abcd.mac
17. abcdmenu.mac
18. abcdn.mac
19. abcds.mac
20. abcmenu.mac
21. abcn.mac
22. abcs.mac
23. abmenu.mac
24. abn.mac
25. abs.mac
26. atom.mac
27. atommenu.mac
28. atomn.mac
29. c2v12.mac
30. c2v12menu.mac
31. c2v12n.mac
32. c2v12s.mac
33. c3v13.mac
34. c3v13menu.mac
35. c3v13n.mac
36. c3v13s.mac
37. d3h13.mac
38. d3h13menu.mac
39. d3h13n.mac
40. d3h13s.mac
41. d4h14.mac
42. d4h14menu.mac
43. d4h14n.mac
44. d4h14s.mac
45. d6h66.mac
46. d6h66menu.mac
47. d6h66n.mac
48. d6h66s.mac
49. ion.mac
50. ionmenu.mac
51. ionn.mac
52. mh2o6.mac
53. mh2o6menu.mac
54. mh2o6n.mac
55. mh2o6s.mac
56. ml4.mac
57. ml4menu.mac
58. ml4n.mac
59. ml4s.mac
60. ml6.mac
61. ml6menu.mac
62. ml6n.mac
63. ml6s.mac
64. oh16.mac
65. oh166.mac
66. oh166menu.mac
67. oh166n.mac
68. oh166s.mac
69. oh16menu.mac
70. oh16n.mac
71. oh16s.mac
72. td14.mac
73. td144.mac
74. td144menu.mac
75. td144n.mac
76. td144s.mac
77. td14menu.mac
78. td14n.mac
79. td14s.mac

4. 講義での使用例

Windows パソコンの基本的な操作は学生全員が習得している．パソコン実習室で一人一台のパソコンを用い，講義を進めていく．本学の学生実習用パソコンは起動するたびにハードディスク内容が初期状態に戻るので，秀丸エディタを他用途で使う場合を想定し，DV–Xα法計算支援環境の C:¥dvxa¥Macros¥SCAT.reg を秀丸エディタで読み込む作業は毎回行う必要がある．秀丸エディタのその他(O)→設定内容の保存/復元(U)…で SCAT.reg を読み込むと，教育用分子軌道計算システム eduDV が実行できる状態になる．

秀丸エディタのウィンドウには，DV–Xα法関係の多種多様なプログラムがボタン（ツールバー，ファンクションキー），プルダウンメニュー, ポップアップメニュー (ユーザーメニュー)，キーボード・ショートカットという形で登録されており，ユーザーは GUI 操作を通じて一連の計算・可視化を実行できる．講義では eduDV ボタンと VESTA ボタンのみを使う（図４）．

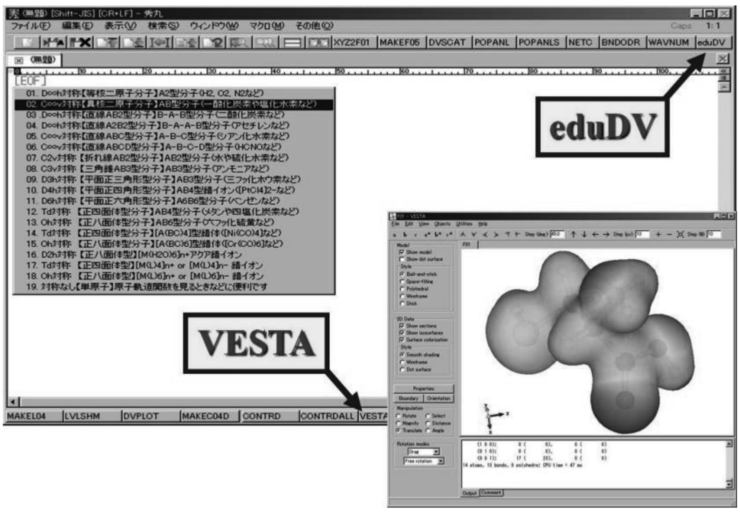

図４　eduDV ボタンと VESTA ボタン

eduDV ボタンを押すと，20 項目のプルダウンメニュー（図５）が現れる．eduDV には，単原子，単原子イオン，分子，錯体，錯イオンの形（点群対称）の違いによる別個のシステムが 20 種類用意されており，システムを選択・起動した後は，必要最低限の情報（原子番号や原子間距離，必要に応じて結合角や酸化数）を会話形式で入力していくだけで，DV−Xα法本体プログラムおよび各種ユーティリティプログラムが全自動で実行される．対称軌道ファイルがシステムに内蔵されているため，分子軌道の縮退もきちんと取り扱われ，分子軌道には教科書と同様の正式な名称が付与される．

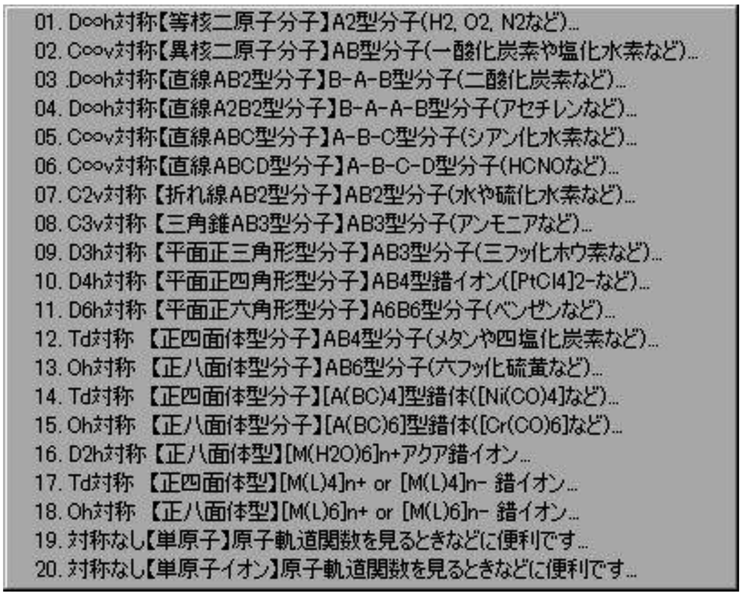
01. D∞h対称【等核二原子分子】A2型分子(H2, O2, N2など)...
02. C∞v対称【異核二原子分子】AB型分子(一酸化炭素や塩化水素など)...
03 .D∞h対称【直線AB2型分子】B-A-B型分子(二酸化炭素など)...
04. D∞h対称【直線A2B2型分子】B-A-A-B型分子(アセチレンなど)...
05. C∞v対称【直線ABC型分子】A-B-C型分子(シアン化水素など)...
06. C∞v対称【直線ABCD型分子】A-B-C-D型分子(HCNOなど)...
07. C2v対称【折れ線AB2型分子】AB2型分子(水や硫化水素など)...
08. C3v対称【三角錐AB3型分子】AB3型分子(アンモニアなど)...
09. D3h対称【平面正三角形型分子】AB3型分子(三フッ化ホウ素など)...
10. D4h対称【平面正四角形型分子】AB4型錯イオン([PtCl4]2-など)...
11. D6h対称【平面正六角形型分子】A6B6型分子(ベンゼンなど)...
12. Td対称 【正四面体型分子】AB4型分子(メタンや四塩化炭素など)...
13. Oh対称 【正八面体型分子】AB6型分子(六フッ化硫黄など)...
14. Td対称 【正四面体型分子】[A(BC)4]型錯体([Ni(CO)4]など)...
15. Oh対称 【正八面体型分子】[A(BC)6]型錯体([Cr(CO)6]など)...
16. D2h対称【正八面体型】[M(H2O)6]n+アクア錯イオン...
17. Td対称 【正四面体型】[M(L)4]n+ or [M(L)4]n- 錯イオン...
18. Oh対称 【正八面体型】[M(L)6]n+ or [M(L)6]n- 錯イオン...
19. 対称なし【単原子】原子軌道関数を見るときなどに便利です...
20. 対称なし【単原子イオン】原子軌道関数を見るときなどに便利です...

図５　eduDV のプルダウンメニュー

VESTA ボタンを押すと，原子群の情報（原子番号と原子座標）に基づいて構造モデルを棒球表示できる．さらに計算した波動関数，電子密度，静電ポテンシャルの三次元ピクセルデータを読み込み，構造モデルに加えて波動関数の等値曲面・断面図や静電ポテンシャルマップ（等電子密度曲面を静電ポテンシャルの大小により彩色した図）などを容易に作成できる．また複数の三次元ピクセルデータを差分・加算表示することも可能で，差電子密度，有効スピン密度などの等値曲面・断面図も描画できる．さらに三次元ピクセルデータを自乗して表示することもできるので，波動関数を自乗することにより電子の発見される確率，すなわち電子密度を計算・表示できる．

5．学生の反応と教育効果

学生からまず上がる声は，波動関数って美しい！という感想である（図６）．周期表の全元素の全原子軌道を容易に計算・三次元可視化できるので，炭素の 2p 軌道(3 種類)，銅の 3d 軌道(5 種類)を手始めに，臭素の 4p 軌道(3 種類)，鉛の 6p 軌道(3 種類)，金の 5d 軌道(5 種類)，ウランの 5f 軌道(7 種類)など，手当たり次第に計算して眺めることができる．

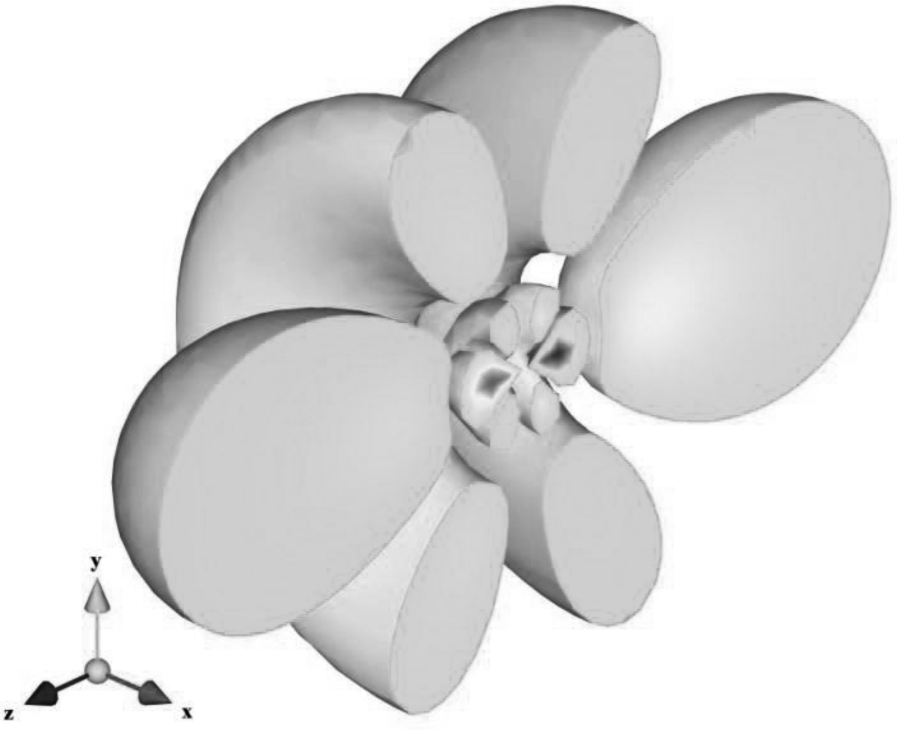

図６　ウランの $5f_{z^3}$ 原子軌道の等値曲面の断面図

次に学生からあがるのは，教科書と同じ結果の分子軌道計算が簡単にできる！という声である．水素分子はもとより，酸素分子，窒素分子，メタン分子，アセチレン分子，水分子，アンモニア分子，ベンゼン分子など，無機化学や物理化学や量子化学の教科書に載っている分子軌道エネルギー準位図と同じ結果が容易に得られる．しかもその分子軌道それぞれについて，VESTA ですべて三次元可視化できる．

例えば岡山理科大学理学部化学科の一年次前期必修講義「コンピュータ入門 I」では，90 分間の講義時間中に，受講生一人一人がまず教育用分子軌道計算システム eduDV の計算環境構築（秀丸エディタにて，eduDV の秀丸エディタマクロ集が利用できるようにする作業）を用い，教育用分子軌道計算システム eduDV を初めて使って，以下の原子・分子の電子状態をすべて自ら計算し，波動関数を VESTA で表示・観察することができた．

1. ネオン原子（$2p_x$ 軌道）
2. アルゴン原子（$3p_x$ 軌道）
3. 水(H_2O)分子（$1b_2$ 軌道(HOMO), $4a_1$ 軌道(LUMO)，静電ポテンシャルマップ）
4. ベンゼン(C_6H_6)分子（$1e_{1g}$ 軌道(HOMO), $1e_{2u}$ 軌道(LUMO)，静電ポテンシャルマップ）
5. 六フッ化ウラン(UF_6)分子（$10t_{1u}$ 軌道(HOMO), $2a_{2u}$ 軌道(LUMO)）

現在の標準的なパソコンは，一昔前の大型計算機に比べて比較にならないほど高性能である．大学にある計算機資源（パソコン）に教育用分子軌道計算システム eduDV その他一式のプログラムを導入することで，基礎化学教育で学生に嫌われがちな量子化学入門を，遊び感覚で正確なイメージの得られる教育内容に変革できる．

参考文献・URL

[1] 荻野博，飛田博実，岡崎雅明, “基本無機化学”，第 2 版，東京化学同人, 2006.
[2] H. Adachi, M. Tsukada, C. Satoko, *J. Phys. Soc. Jpn.*, **45**, 875 (1978).
[3] 足立裕彦監修, “はじめての電子状態計算”，三共出版, 1998.
[4] http://chem.sci.hyogo-u.ac.jp/hajimete/download.html
[5] http://hide.maruo.co.jp/software/hidemaru.html
[6] http://homepage.mac.com/fujioizumi/visualization/VENUS.html#assistance_environment
[7] G. Sakane, K. Momma, F. Izumi, *Bull. Soc. DV–Xα*, **21**(1&2), 13 (2008).
[8] K. Momma, F. Izumi, *J. Appl. Crystallogr.*, **41**, 653 (2008).
[9] http://www.geocities.jp/kmo_mma/crystal/jp/vesta.html
[10] 坂根弦太, *日本教育情報学会第 22 回年会（岡山）論文集*, 2D3, 198 (2006).
[11] http://www.chem.ous.ac.jp/%7Egsakane/HidemaruDV/HidemaruDV.pdf
[12] 坂根弦太, *JUCE Journal（大学教育と情報）*, **18** (4), 15 (2010), http://www.juce.jp/LINK/journal/1002/03_03.html
[13] http://www.chem.ous.ac.jp/%7Egsakane/fun/index.html#edudv
[14] http://www.openwatcom.org/

複数制約 0−1 ナップザック問題に対するグリーディ法の適用

情報処理センター

岩崎彰典

１．はじめに

0-1 ナップザック問題は N-P 困難な問題である．グリーディ法は比較的良好な近似解を求める方法であり，そこで求められた下限値と上限値を使って分枝限定法は厳密解を求める方法である。ここでは，従来扱われてきた単一制約 0-1 ナップザック問題を複数制約 0-1 ナップザック問題へ拡張し近似解を求めることを試みる．

２．複数制約 0-1 ナップザック問題

N個の要素と M個の制約条件を持つ複数制約 0-1 ナップザック問題は次のように定式化される．

$$\text{maxmize} \quad f(\boldsymbol{x}) = \sum_{i=1}^{N} f_i\, x_i$$

$$\text{subject to} \quad g_j(\boldsymbol{x}) = \sum_{i=1}^{M} g_{ij} x_i \le b_j$$

$$x_i = 0 \text{ or } 1, \quad i = 1, 2, \dots, N \quad j = 1, 2, \dots, M$$

ここで　$f(x)$ は目的関数，$g(x)$は制約関数，f_i とg_{ij}はそれぞれi番目の要素の価値と制約条件を決める制約量，b_j はj番目の制約許容量である．

ナップザック問題の困難さは変数の離散性に起因している．単一制約で変数が 0 から 1 までの連続値を許す場合は多項式時間のオーダーで厳密解を求めることができ，その値は変数が離散値の場合の近似解となる．このときの手法はf_i/g_iの大きい順番に並べ替え，その大きい要素から変数を 1 に固定する方法である．

すなわち要素の価値と制約量の比が選択の基準となるが，複数制約の場合，制約量を 1 つの量に変換し並べ替えの基準を決定することが必要となる．

我々は 2 制約条件の場合，2 つの制約量に重みをつけ，2 つの制約量を次のように 1 つの制約量g^*に変換する．

$$\alpha_i = \frac{g_{i1}}{g_{i1} + g_{i2}}$$

$$g_i^* = \sqrt{(\alpha_i g_{i1})^2 + ((1 - \alpha_i) g_{i2})^2}$$

３．計算機実験

実験では 100 変数の要素の価値と制約量をランダムに作成した．２つの制約許容量を定めた１つの問題に対し，各要素の７つの並べ替えの基準を設定し実験を行った．

１．$\dfrac{f_i}{g_{i1}}$

２．$\dfrac{f_i}{g_{i2}}$

３．$\dfrac{f_i}{\sqrt{g_{i1}^2+g_{i2}^2}}$

４．暫定解まで $\dfrac{f_i}{\sqrt{g_{i1}^2+g_{i2}^2}}$ を用いる．その後 $\dfrac{f_i}{g_{i1}}$ を用いる．

５．暫定解まで $\dfrac{f_i}{\sqrt{g_{i1}^2+g_{i2}^2}}$ を用いる．その後 $\dfrac{f_i}{g_{i2}}$ を用いる．

６．$\dfrac{f_i}{g_i^*}$

７．暫定解まで $\dfrac{f_i}{\sqrt{g_{i1}^2+g_{i2}^2}}$ を用いる．その後 g_i^* を用いる．

結果を表１に示す．f1～f7 はそれぞれ７つの並べ替えの基準を採用したときの目的関数値である．太字はその中での最良解を示す．

４．まとめ

一般に問題が複数の制約条件を持つときは，その制約の強さには偏りがある．その偏りを補正しつつグリーディー法を適用することにより，高速により良い解を得ることができる．現在は並べ替えの基準を手探りで決め，最良の解を探しているが，今後はコンピュータが問題の上限値を基に，自動的に並べ替えの基準を変えていくようにしたい．そのためにはより良い上限値が必要であり，それを求めるアルゴリズムの開発も必要である．

５．参考文献

1. 整数計画法と組合せ最適化，今野浩，鈴木久敏，日科技連出版社，１９８２
2. 最適化プログラミング，茨木俊秀，福島雅夫，岩波書店，１９９１

表１　制約許容量と並べ替えの基準による最良解

制約許容量１	制約許容量２	f1	f2	f3	f4	f5	f6	f7
100	5000	691	267	500	500	500	**697**	691
300	4800	1341	457	998	1018	1018	**1352**	1341
500	4600	**1781**	595	1416	1416	1416	**1781**	**1781**
700	4400	2133	997	1857	1888	1888	**2134**	2133
900	4200	2452	1295	2246	2270	2270	**2453**	2452
1100	4000	**2718**	1645	2280	2526	2500	2715	**2718**
1300	3800	**2945**	1710	2550	2604	2608	2940	**2945**
1500	3600	3160	1924	2759	2894	2894	**3169**	3160
1700	3400	3358	2196	2978	3007	3007	3347	**3363**
1900	3200	**3560**	2551	3144	3251	3251	3547	3548
2100	3000	3425	2997	3449	2449	3449	**3682**	3678
2300	2800	2997	3263	3741	3749	3749	**3793**	3775
2500	2600	2891	3485	**3771**	3719	**3771**	**3771**	3762
2700	2400	2600	**3685**	3628	3574	3628	3628	**3685**
2900	2200	2461	**3558**	3438	3438	3451	3513	**3558**
3100	2000	2304	3389	3303	3315	3360	3374	**3391**
3300	1800	1951	**3226**	3167	3098	3167	3204	**3226**
3500	1600	1818	**3028**	2926	2837	2917	3007	**3028**
3700	1400	1701	**2804**	2615	2615	2615	2812	**2804**
3900	1200	1470	**2597**	2336	2259	2332	**2597**	**2597**
4100	1000	1305	2336	1930	1930	1910	**2353**	2336
4300	800	1081	**2124**	1571	1571	1761	**2124**	**2124**
4500	600	861	**1797**	1501	1273	1557	1792	**1797**
4700	400	592	**1430**	1249	1021	1325	**1430**	**1430**

Sharma の８次の反復式による数列の収束加速法の提案

総合情報学部・情報科学科　榊原道夫

１．はじめに

収束数列の加速法として Aitken δ²法がよく知られている。この加速法の統計計算への応用の１例は黒田、榊原によりＥM反復法の加速として与えられている[1]。Aitken δ²法は収束列$\{x_k\}$に対して変換

（１）
$$y_k = x_k - \frac{(x_{k+1}-x_k)^2}{x_{k+1}-2x_k+x_{k-1}}$$

により与えられる。$\{y_k\}$は$\{x_k\}$より速く集積点x^*に収束する。$\{x_k\}$が反復式

（２）
$$x_{k+1} = F(x_k)$$

により与えられるとき（１）は方程式

（３）
$$f(x) := x - F(x) = 0$$

に対して Secant 法を適用し得られる反復式の右辺に一致する。この定式化の形式化は多くの提案されている方程式（３）に対する反復法の数列加速への適用方法与える。ここでは、高次の反復法で８次の場合のいくつかの例を適用した場合を考察する。

２．８次の反復法

J.R.Sharma and R.Sharma[2]による８次の反復式は

（４－１）
$$w_k = x_k - N(x_k)$$

（４－２）
$$z_k = w_k - \frac{f(x_k)}{f(x_k)-2f(w_k)}N(x_k)$$

（４－３）
$$x_{k+1} = z_k - \left[1+\frac{f(z_k)}{f(x_k)}+\alpha\left(\frac{f(z_k)}{f(x_k)}\right)^2\right]\frac{f[x_k,w_k]f(z_k)}{f[w_k,z_k]f[x_k,z_k]}$$

である。ここでαは任意の実数で、

（５）　$$N(x)=\frac{f(x)}{f'(x)},\quad f[a,b]=\frac{f(b)-f(a)}{b-a}$$

である。この反復法は８次収束である。αは任意の実数でよいので$\alpha=0$と取ることにより

（６）　$$x_{k+1}=z_k-\left[1+\frac{f(z_k)}{f(x_k)}\right]\frac{f[x_k,w_k]f(z_k)}{f[w_k,z_k]f[x_k,z_k]}$$

を得る。（４－１，２）＋（６）がこの反復法で最も簡単な場合である。

３．新たな加速法

（４－１，２）＋（６）より加速式を与えるために$N(x)$を Secant 法により与えられる作用素

（７）　$$S(a,b):=\frac{f(b)}{f[a,b]}=\frac{b-a}{f(b)-f(a)}f(b)$$

に置き換える。今ことより次の加速式：

（８－１）　$$W_k=x_k-S(x_k,x_{k-1})$$

（８－２）　$$Z_k=W_k-\frac{f(x_k)}{f(x_k)-2f(W_k)}S(x_k,x_{k-1})$$

（８－３）　$$x_{k+1}=z_k-\left[1+\frac{f(Z_k)}{f(x_k)}\right]\frac{f[x_k,W_k]f(Z_k)}{f[W_k,Z_k]f[x_k,Z_k]}$$

を得る。この変換を収束列$\{x_k\}$について書き直すと

（９－１）　$$W_k=x_k-\frac{(x_{k+1}-x_k)^2}{x_{k+1}-2x_k+x_{k-1}}$$

（９－２）　$$Z_k=W_k-\frac{f(x_k)}{x_k-x_{k+1}-2(W_k-F(W_k))}\frac{(x_{k+1}-x_k)^2}{x_{k+1}-2x_k+x_{k-1}}$$

（９－３）　$$x_{k+1}=z_k-\left[1+\frac{Z_k-F(Z_k)}{x_k-x_{k+1}}\right]\frac{f[x_k,W_k](Z_k-F(Z_k))}{f[W_k,Z_k]f[x_k,Z_k]}$$

となる。また、

（１０－１）　$$f[x_k,W_k]=\frac{W_k-F(W_k)-x_k+x_{k+1}}{W_k-x_k}$$

（１０－２）　$$f[x_k,Z_k]=\frac{Z_k-F(Z_k)-x_k+x_{k+1}}{Z_k-x_k}$$

（１０－３）　$$f[W_k,Z_k]=\frac{Z_k-F(Z_k)-W_k+F(W_k)}{Z_k-W_k}$$

である。そのため、$F(W_k), F(Z_k)$の評価が必要となる。

４．数値例

数値例を問題

（１１）　$$x-(x)^{29/30}=0$$

を取り上げる。この方程式の解は$x=1$である。初期値$x_0=1000$として反復式

（１２）　$$x_{k+1}=(x_k)^{29/30}$$

により生成される収束列$\{x_k\}$に対して加速の結果を表１に示す。

表１

方法	（１２）の反復	Aitken δ^2法	提案する方法
反復回数	１０７６	５５９	２０５
ＣＰＵ時間（秒）	０．８４３	０．５１５	０．４３７

表１より提案する方法は、（１２）の約１／４、Aitken δ^2法の約半分である。ＣＰＵ時間でも優越した結果となった。

５．まとめ

Sharmaの８次の反復法を用いた加速法の提案を示した。数値例より加速法として十分な性能があることがわかった。特に加速のために２つのプロセスがAitken δ^2法よりも多くなっているにもかかわらずCPU時間の比較で優れている点が提案する方法の良い点である。

参考文献

[1] Kuroda,M. and Sakakihara, M., Acceleration of the EM and ECM algorithms using the Aitken delta^2 method for log-linear models with partially classified data, Statistics and Probability Letters, 78, pp. 2332 – 2338, 2008.

[2] Sharma, J.R. and Sharma, R., A new family of modified Ostrowski's methods with accelerated eighth order convergence, Numer Algor, 54, pp445-458, 2010.

微分方程式法による代数曲線の描画

榊原道夫（岡山理科大学・総合情報学科・情報科学科）
本淨清佳（2008 年度榊原ゼミ卒業生）

1　はじめに

本報告において我々は、微分方程式の数値解法を用いて代数曲線を描画するアルゴリズムについて考察する。計算数学ツール Maple を用い代数曲線を描画する場合 Maple がもつ impliciteplot 関数が存在する。しかし、実際に Maple を使い描画させる。数式を描画するにあたり、グラフの線に誤差が生じてくる。そのために初期設定の所で正しいグラフになるような値を入力することで、部分分け結果の図が上手く描画でき、その図で生じた誤差を最小限に抑えることができる。また、部分分けプログラムと implicitplot の描画結果を比較してどちらの方法が上手く描画できているかを見る。implicitplot は部分分けプログラムと違い、プログラム内容としてはとても簡単なプログラムである。その 2 つのプログラムを比較することにより上手く描画し、尚且つ効率の良い方法を得ることを目的とする。

2　微分方程式法による描画法

2 変数の任意の陰関数 F(x, y) = 0 について考える。この関数を x で微分すると、

$$\frac{dy}{dx} = -\frac{\frac{\partial F}{\partial x}}{\frac{\partial F}{\partial y}} = g(x, y)$$

である。また、簡単な微分方程式の初期値問題の解法としてオイラー法がある。しかし、オイラー法は簡単であるけれども、とても細かな時間刻み幅を用いないと十分な精度が出てこないが、この方法を用いると、

$$x_{i+1} = x_i + \Delta x$$
$$y_{i+1} = y_i + g(x_i, y_i)\Delta x$$

の数値解が得られる。ここで Δx は時間刻み幅であり、i = 0 , 1 , 2 . . . である。上記の式は y が x の 1 価単調連続な関数の時に計算することができる。y が x の多価の単調で連続な関数の場合に拡張するには、$g(x, y)$ の絶対値がある境目となる値 G より大きくなったときに積分の軸を x から y に変更すればよい。つまり $|g(x, y)| > G$ で計算すればよい。この時の軸を変更した式は、

$$y_{l+1} = y_l + \Delta y$$

$$x_{l+1} = x_l + \xi_l \Delta y / g(x_l, y_l)$$

となる。ここで、ξ_1の範囲は、

$$\xi_l = 1 \quad (x_l > x_{l-1})$$

$$\xi_l = -1 \quad (x_l < x_{l-1})$$

とする。ξ_1を用いたときの、$|g(x,y)| < G$の場合は次のようになる。

$$x_{i+1} = x_{i+1} + \xi_{i+1} \Delta x$$

$$y_{i+1} = y_{i+1} + \xi_{i+1} g(x_{i+1}, y_{i+1}) \Delta x$$

上記の２つの場合の式を交互に使い分けることにより、y が x の多価の単調で連続な関数の場合の数値を求めることができる。　また、自己作成したプログラム中にはΔxをh, Δy をmと表現している。hやmの幅を大きくしたり、小さくすることで線の誤差が生じたり綺麗に描画できるようになる。

先ほどの、$g(x,y)$の絶対値がある境目となる値G について $x^2 + y^2 - 1 = 0$ の数式を用いて説明してみる。図1.は初期値x=0.0および-1.0　Δx =0.02　G =1.0の場合を計算している。先ほども言ったように数値計算上の誤差のために円が閉じていない状態である。しかし、Δx を十分に小さくすれば正しい円になる。G の依存を調べるためにG =10.0と100.0の場合を計算したが、どちらもほぼ同じ結果になった。G の違いの誤差というのは、オイラー法の数値計算上の誤差に比べてみるとほぼ1桁以上小さいものである。これは微係数dy/dxの変化が $x = 1.0$ と -1.0 辺りで変化が生じてるためと思われる。数値計算上では発散する部分では、軸が変わっていて y 軸方向に積分が行われているので $x = 1.0$ または -1.0 辺りのオーバーフローは起こらないのである。このような現象を抑えるためには場合分けをする必要がある。場合分けをする場合のx, yの初期値設定について提案する。初期値の設定は、描画する計算式の極限値を求めておく。そうすることにより1つの値に収束されていき値が求められ、それから画範囲が決めることができるので描画できる。

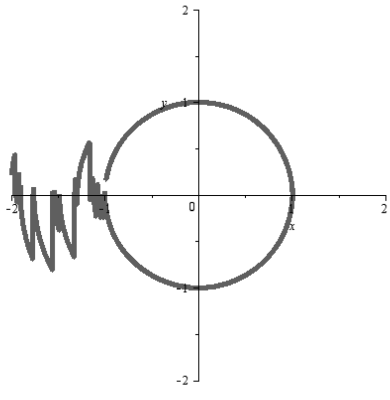

図 1.　初期値 x=0.0,-1.0　　Δx =0.02 の場合を表示する。

3　Mapleによる代数曲線の描画例

3.1計算例［$x^4 + y^4 - 2y(y^2 - x^2) = 0$］

$$x^4 + y^4 - 2y(y^2 - x^2) = 0$$

のグラフの y は、1 価から 4 価の関数となる。この式を微分すると、

$$dy/dx = g(x, y) = -(2y^3 + 2xy)/(2x^3 - 3x^2 + y^2)$$

である。実際にMapleを使いグラフを描いてみる。

プログラム 3.1　部分分けプログラム

```
for i from 1 to m do
x[i]:=x[i-1]+h;
y[i]:=y[i-1]+h*subs({a=x[i-1],b=y[i-1]},g);
end do:
l1 := [[x[n],y[n]] $n=1..m]:
p1:=plot(l1, x=-1..1,y=-0.6..2, style=line,symbol=circle):
```

上記はプログラムの一部であり、初期値$x = 0.0$　$y = 2.0$の場合を計算している。このプログラム 3.1 は第 2 象限を表示させる。For 文では m＝715 としているので 1 から 715 までの繰り返しを計算している。初期値の時にΔx を h=0.001 と設定しているので図 2.の線の範囲まで描画できるようになっている。例えば h の部分のところに他の数値を入力すると図の線がオーバーフローを起こしたり、線が途中で途切れてしまってしまう。style=line,symbol=circle の部分はグラフの線のスタイルを設定している。

プログラム 3.2　部分分けプログラム

```
for i from 1 to m do
x[i]:=x[i-1]-h;
y[i]:=y[i-1]-h*subs({a=x[i-1],b=y[i-1]},g);
end do:
l2 := [[x[n],y[n]] $n=1..m]:
p2:=plot(l2, x=-1..1,y=-0.6..2, style=line,symbol=circle):
```

初期値$x = 0.0$　$y = 2.0$の場合を計算している。このプログラムは第 1 象限を表示させる。For 文では m＝715 としているので 1 から 715 までの繰り返しを計算している。初期値の時にΔx を h=0.001 と設定しているので図 2.の線の範囲まで描画できるようになっている。例えば h の部分のところに他の数値を入力すると図の線がオーバーフローを起こしたり、線が途中で途切れてしまってしまう。この 2 つのプログラム 3.1 と 3.2 を表示させたものが図 2.である。

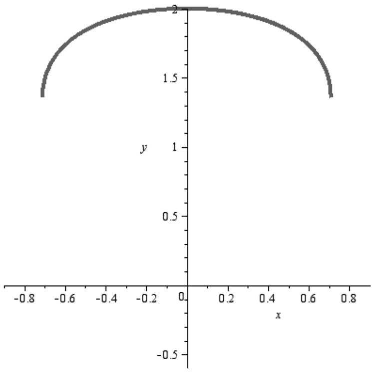

図 2. 初期値 $x = 0.0$　 $y = 2.0$ の実行結果

プログラム 3.3　部分分けプログラム

```
F:=a^4+b^4-2*a*(a^2-b^2);
x[0]:=0.0;
y[0]:=0.5;
for i from 1 to m do
x[i]:=x[i-1]+h;
y[i]:=y[i-1]+h*subs({a=x[i-1],b=y[i-1]},g);
end do:

l3:= [[y[n],x[n]] $n=1..m]:
p3:=plot(l3, x=-1..1,y=-0.6..2, style=line,symbol=circle):
```

初期値 $x = 0.0$　 $y = 0.5$ の場合を計算している。このプログラム 3.3 は第 2 象限を表示させる。上記のプログラム 3.3 は始めに F の部分で a と b を入れ替えている。これは x 軸と y 軸を入れ替えることによって初期値を与えやすく、計算し易くしている。For 文では m＝1315 としているので 1 から 1315 までの繰り返しを計算している。初期値の時に Δx を h=0.001 と設定しているので図 3.の線の範囲まで描画できるようになっている。例えば h の部分のところに他の数値を入力すると図の線がオーバーフローを起こしたり、線が途中で途切れてしまってしまう。表示するときは、プログラムの l3 で x 軸と y 軸を入れ替えて元に戻す。

プログラム 3.4　部分分けプログラム

```
F:=a^4+b^4-2*a*(a^2-b^2);

x[0]:=0.0;
y[0]:=-0.5;
for i from 1 to m do
x[i]:=x[i-1]+h;
y[i]:=y[i-1]+h*subs({a=x[i-1],b=y[i-1]},g);
end do:
```

```
l4:= [[y[n],x[n]] $n=1..m]:
p4:=plot(l4, x=-1..1,y=-0.6..2, style=line,symbol=circle):
display({p1,p2,p3,p4});
```

先ほどのプログラム 3.3 と同様に初期値 $x = 0.0 \quad y = -0.5$ の場合を計算している。このプログラム 3.4 は第 2 象限を表示させる。For 文では m＝1315 としているので 1 から 1315 までの繰り返しを計算している。初期値の時に Δx を h=0.001 と設定しているので図 3.の線の範囲まで描画できるようになっている。例えば h の部分のところに他の数値を入力すると図の線がオーバーフローを起こしたり、線が途中で途切れてしまってしまう。この 4 つのプログラム 3.1 と 3.2 と 3.3 と 3.4 を表示させたものが図 3.である。

初期値 $x = 0.0 \quad y = 0.5$ の場合を計算する。このプログラム 3.5 は第 3 象限を表示させる。上記のプログラムは始めに F の部分で a と b を入れ替えている。これは x 軸と y 軸を入れ替えることによって初期値を与えやすく、計算し易くしている。For 文では m＝515 としているので 1 から 515 までの繰り返しを計算している。初期値の時に Δx を h=0.001 と設定しているので図 4.の線の範囲まで描画できるようになっている。例えば h の部分のところに他の数値を入力すると図の線がオーバーフローを起こしたり、線が途中で途切れてしまってしまう。表示するときは、プログラムの l5 で x 軸と y 軸を入れ替えて元に戻す。

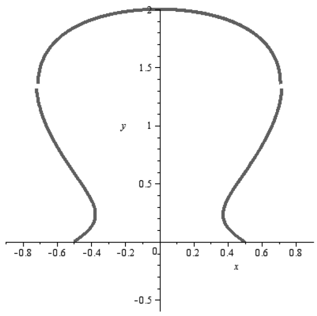

図 3.　初期値 $x = 0.0, y = 0.5$ と $x = 0.0, y = -0.5$ の実行結果

プログラム 3.5　部分分けプログラム

```
F:=a^4+b^4-2*a*(a^2-b^2);
x[0]:=0.0;
y[0]:=0.5;
for i from 1 to m do
x[i]:=x[i-1]-h;
y[i]:=y[i-1]-h*subs({a=x[i-1],b=y[i-1]},g);
end do:
l5:= [[y[n],x[n]] $n=1..m]:
p5:=plot(l5, x=-1..1,y=-0.6..2, style=line,symbol=circle):
```

プログラム 3.6　部分分けプログラム

```
x[0]:=0.0;
y[0]:=-0.5;
for i from 1 to m do
x[i]:=x[i-1]-h;
y[i]:=y[i-1]-h*subs({a=x[i-1],b=y[i-1]},g);
end do:
l6:= [[y[n],x[n]] $n=1..m]:
p6:=plot(l6, x=-1..1,y=-0.6..2, style=line,symbol=circle):
```

先ほどのプログラムと同様に初期値 $x = 0.0 \quad y = -0.5$ の場合を計算している。このプログラム 3.6 は第 4 象限を表示させる。For 文では m＝515 としているので 1 から 515 までの繰り返しを計算している。初期値の時に Δx を h=0.001 と設定しているので図 4.の線の範囲まで描画できるようになっている。例えば h の部分のところに他の数値を入力すると図の線がオーバーフローを起こしたり、線が途中で途切れてしまってしまう。この 6 つのプログラム 3.1 から 3.6 を表示させたものが図 4.である。

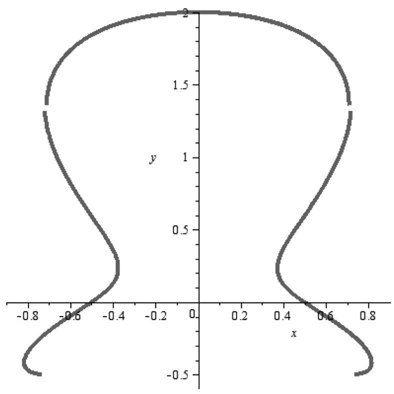

図 4. 初期値 $x = 0.0, y = 0.5$ と $x = 0.0, y = -0.5$ の実行結果

プログラム 3.7　部分分けプログラム

```
x[0]:=0.5;
y[0]:=-0.5;
for i from 1 to m do
x[i]:=x[i-1]-h;
y[i]:=y[i-1]-h*subs({a=x[i-1],b=y[i-1]},g);
end do:
l7 := [[x[n],y[n]] $n=1..m]:
p7:=plot(l7, x=-1..1,y=-0.6..2, style=line,symbol=circle):
```

上記はプログラムの一部であり、初期値 $x = 0.5 \quad y = -0.5$ の場合を計算している。For 文では m＝1200 としているので 1 から 1200 までの繰り返しを計算している。初期値の時に Δx を h=0.001 と設定しているので図 5.の線の範囲まで描画できるようになっている。例えば h の部分のところに他の数値を入力すると図の線がオーバーフローを起し、線が途中で途切れてしまってしまう。このプログラム 3.7 は第 3

象限を表示させる。

プログラム 3.8　部分分けプログラム

```
> x[0]:=-0.5;
y[0]:=-0.5;
for i from 1 to m do
x[i]:=x[i-1]+h;
y[i]:=y[i-1]+h*subs({a=x[i-1],b=y[i-1]},g);
end do:
l8 := [[x[n],y[n]] $n=1..m]:
p8:=plot(l8, x=-1..1,y=-0.6..2, style=line,symbol=circle):
```

先ほどのプログラム 3.3 と同様に初期値$x=0.5$　$y=-0.5$の場合を計算している。このプログラム 3.8 は第 4 象限を表示させる。For 文では m＝1200 としているので 1 から 1200 までの繰り返しを計算している。初期値の時にΔxを h=0.001 と設定しているので図 5.の線の範囲まで描画できるようになっている。例えば h の部分のところに他の数値を入力すると図の線がオーバーフローを起こしたり、線が途中で途切れてしまってしまう。この 8 つのプログラム 3.1 から 3.8 を表示させたものが図 5.である。図 5. は今までのプログラムを全て合わして表示させた図である。

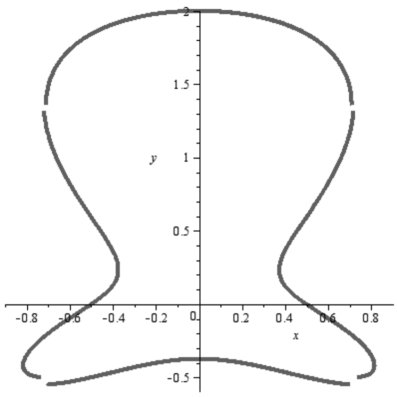

図 5. 初期値$x=0.5$　$y=-0.5$の実行結果

また、implicitplot で上記の数式を描画してみる。どちらの方法が上手く描画できるかを調べる。implicitplot プログラムの内容は先ほどの部分分けプログラムよりもとても簡単に作れることができる。しかし、プログラムが簡単に作っているためにきれいな図にはなりにくい。

プログラム 3.9

```
implicitplot(x^4+y^4-2*y*(y^2-x^2) = 0, x = -1 .. 1, y = -2 .. 2)
```

x, yの値を同じように設定し、その結果が次のようになる。

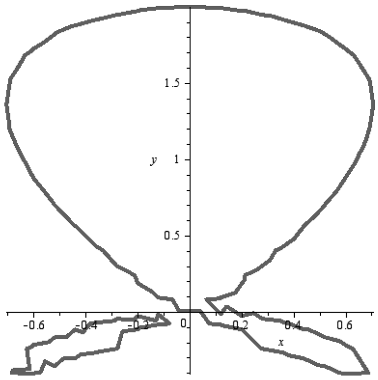

図 6. implicitplot 実行結果

図 5.と図 6.を比較したら、一目で見てわかるようにこの場合は図 6. implicitplot より図 5.の部分分けプログラムで作成する方法で描画するほうが上手く描けることが分かる。図 6. implicitplot では線が繋がっていない部分がある。これは、この数式の図ではないことが分かる。実際の図が図 5.の部分分けプログラムである。

3.2 計算例 [$x^2-y^2-(x^2+y^2)^2=0$]

$x^2-y^2-(x^2+y^2)^2=0$ を描画する。Maple のプログラム 4.1 を用いグラフを描く。

プログラム 4.1

```
restart;
with(plots);
Digits := 12;
m := 300;
h := 0.005;
F := a^2-b^2-(a^2+b^2)^2;
g := -(diff(F, a))/(diff(F, b));
x := array(0 .. m);
y := array(0 .. m);
eq := subs(a = 0, F); sol := solve(eq = 0, b);
x[0] := 0.5;
eq := subs(a = x[0], F); sols := solve(eq = 0, b); y[0] := sols[4]
```

$$eq := 0.25 - b^2 - \left(0.25 + b^2\right)^2$$

$$sols := -1.27122987842\,I,\ 1.27122987842\,I,\ -0.340625019317,\ 0.340625019317$$

$$y_0 := 0.340625019317$$

初期設定で $eq := subs(a = x[0], F); sols := solve(eq = 0, b); y[0] := sols[4];$ と設定している。これは始め

にxの値を決めてからその値で計算した結果、yの値を設定するようにしているものである。そうすることにより、グラフの線の上にある数x, yを見つけるためである。$sols[4]$というのはプログラム中に計算結果として表示されている$sols$の値で前から4番目の数をyの値としている。

プログラム 4.2

```
> for i to m do
x[i] ≔ x[i-1]-h;
y[i] ≔ y[i-1]-h*subs({a = x[i-1], b = y[i-1]}, g)
end do;
l1 ≔ [`$`([x[n], y[n]], n = 1 .. m)];
p1 ≔ plot(l1, x = -1 .. 1, y = -0.4 .. 0.4, style = line, symbol = circle);
```

上記はプログラムの一部であり、初期値$x = 0.5$　$y = 0.3$の場合を計算している。For 文では m＝300 としているので 1 から 300 までの繰り返しを計算している。初期値の時にΔxを h=0.005 と設定しているので図 7.の線の範囲まで描画できるようになっている。例えば h の部分のところに他の数値を入力すると図の線がオーバーフローを起こし、線が途中で途切れてしまってしまう。このプログラム 4.2 を表示させたのが図 7.である。

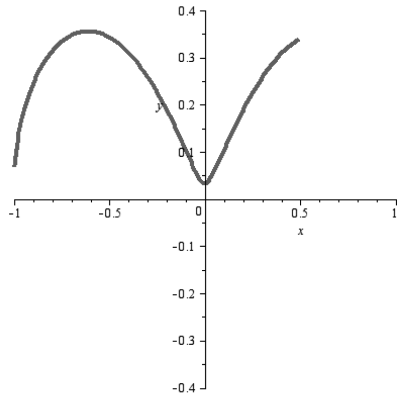

図 7. 初期値$x = 0.5$　$y = 0.3$の実行結果

プログラム 4.3

```
x[0] ≔ -0.5;
eq ≔ subs(a = x[0], F); sols ≔ solve(eq = 0, b); y[0] ≔ sols[3];
for i to m do
x[i] ≔ x[i-1]+h;
y[i] ≔ y[i-1]+h*subs({a = x[i-1], b = y[i-1]}, g)
end do;
l2 ≔ [`$`([x[n], y[n]], n = 1 .. m)];
p2 ≔ plot(l2, x = -1 .. 1, y = -0.4 .. 0.4, style = line, symbol = circle)
```

$$x_0 := -0.5$$

$$eq := 0.25 - b^2 - \left(0.25 + b^2\right)^2$$

$$sols := -1.27122987842\,I, 1.27122987842\,I, -0.340625019317, 0.340625019317$$

$$y_0 := -0.340625019317$$

初期設定で$eq := subs(a = x[0], F); sols := solve(eq = 0, b); y[0] := sols[3];$と設定している。これは始めに x の値を決めてから y の値を設定するようにしているものである。そうすることにより、グラフの線の上にある数x, yを見つけるためである。$sols[3]$というのはプログラム中に表示されている$sols$の値で前から 3 番目の数を $y[0]$ の値としている。このプログラム 4.3 を表示させたものが図 8.である。

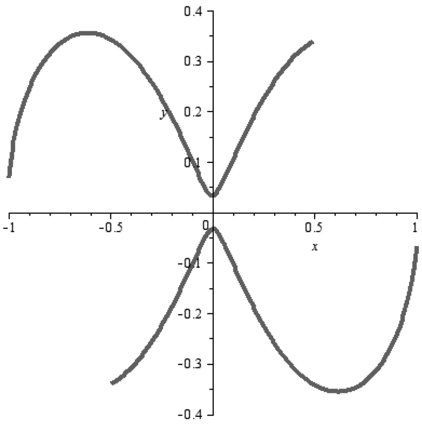

図 8. 初期値$x = -0.5$　$y = -0.3$の実行結果

プログラム 4.4

```
x[0] ≔ -0.5;
eq ≔ subs(a = x[0], F); sols ≔ solve(eq = 0, b); y[0] ≔ sols[3];
for i to m do
x[0] ≔ -0.5;
eq ≔ subs(a = x[0], F); sols ≔ solve(eq = 0, b); y[0] ≔ sols[3];
   for i to m do
     x[i] ≔ x[i-1]-h;
     y[i] ≔ y[i-1]-h*subs({a = x[i-1], b = y[i-1]}, g)
   end do;
end do;
l3 ≔ [`$`([x[n], y[n]], n = 1 .. m)];
p3 ≔ plot(l3, x = -1 .. 1, y = -0.4 .. 0.4, style = line, symbol = circle)
```

$$x_0 := -0.5$$

$$eq := 0.25 - b^2 - \left(0.25 + b^2\right)^2$$

$$sols := -1.27122987842\,\mathrm{I},\ 1.27122987842\,\mathrm{I},\ -0.340625019317,$$
$$0.340625019317$$

$$y_0 := -0.340625019317$$

初期設定で $eq := subs(a = x[0], F); sols := solve(eq = 0, b); y[0] := sols[3];$ と設定している。これは始めに x の値を決めてから y の値を設定するようにしているものである。そうすることにより、グラフの線の上にある数 x, y を見つけるためである。$sols[3]$ というのはプログラム中に表示されている $sols$ の値で前から 3 番目の数を $y[0]$ の値としている。このプログラム 4.4 を表示させたものが図 9.である。

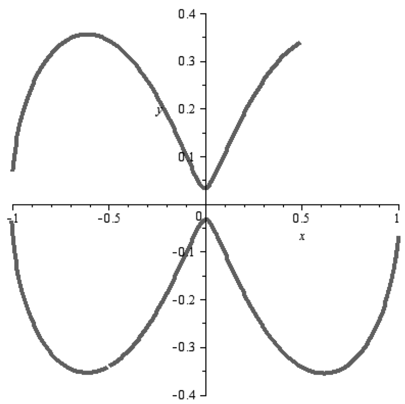

図 9. 初期値 $x = -0.5$　$y = -0.3$ の実行結果

プログラム 4.5

```
x[0] := 0.5;
eq := subs(a = x[0], F); sols := solve(eq = 0, b); y[0] := sols[4]
```

$$x_0 := 0.5$$

$$eq := 0.25 - b^2 - \left(0.25 + b^2\right)^2$$

$$sols := -1.27122987842\,\mathrm{I},\ 1.27122987842\,\mathrm{I},\ -0.340625019317,$$
$$0.340625019317$$

$$y_0 := 0.340625019317$$

```
for i to m do
x[i] := x[i-1]+h;
y[i] := y[i-1]+h*subs({a = x[i-1], b = y[i-1]}, g)
end do;
l4 := [`$`([x[n], y[n]], n = 1 .. m)];
p4 := plot(l4, x = -1 .. 1, y = -0.4 .. 0.4, style = line, symbol = circle);
display({p1, p2, p3, p4})
```

初期設定で $eq := subs(a = x[0], F); sols := solve(eq = 0, b); y[0] := sols[4];$ と設定している。これは始めに x の値を決めてから y の値を設定するようにしているものである。そうすることにより、グラフの線の上にある数 x, y を見つけるためである。$sols[4]$ というのはプログラム中に表示されている $sols$ の値で前から 4 番目の数を $y[0]$ の値としている。このプログラム 4.5 を表示させたものが図 10.である。

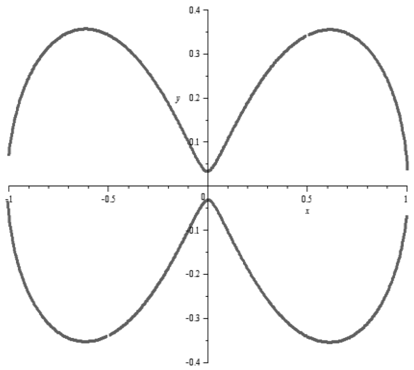

図 10. 初期値 $x = 0.5$　 $y = 0.3$ の実行結果

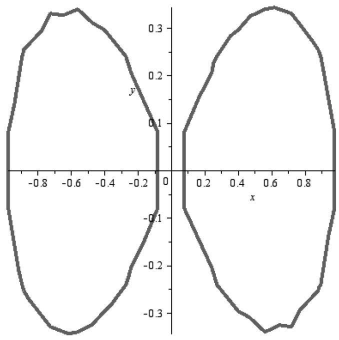

図 11. implicitplot 実行結果

また、implicitplot で上記の数式を描画してみる。図 10.と図 11.を比較したら、一目で見てわかるようにこの場合は図 11. implicitplot より.図 10.の部分分けプログラムで作成する方法で描画するほうが上手く描けることが分かる。図 11.では $x, y = 0$ の部分を見てみると交わっている線になるはずだが、交わっていないことが分かる。図 3.の 0 付近に注目してみると繋がっていない。これは、はじめに $m = 300$ と設定しているため、初期値 $x = 0.5$ から描画しはじめて 0 に近づくにつれて誤差が生じているからだ。$m = 200$ と設定してみると繋がった図が完成する。細かく計算しても、このような場合には誤差が生じるようにもなってくることが分かる。

3.3　計算例［$(x^2 + y^2)^3 - 4x^2y^2 = 0$］

次に、同様に $(x^2 + y^2)^3 - 4x^2y^2 = 0$ の計算式を描画する。。

先ほどの $x^2 - y^2 - (x^2 + y^2)^2 = 0$ と同じ様にプログラムを作成する。そうすると次のような図 12.が得られる。

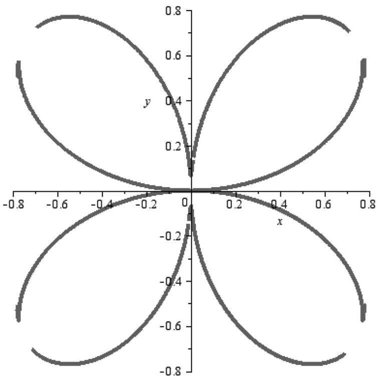

図 12. $(x^2 + y^2)^3 - 4x^2y^2 = 0$ の実行結果

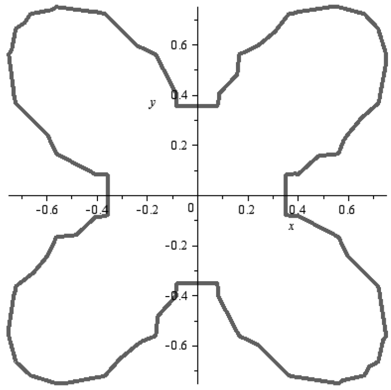

図 13. implicitplot 実行結果

また、implicitplot で上記の数式を描画してみる。どちらの方法が上手く描画できるかを調べる。図 12.と図 13.を比較したら、一目で見てわかるようにこの場合は図 13. implicitplot より.図 12.の部分分けプログラムで作成する方法で描画するほうが上手く描けることが分かる。図 13.では $x, y = 0$ の部分が交わっていない状態であり、全くと言っていいくらい違うグラフになっている。

3.4　計算例［$(x^2+y^2)^2-2x^2-y^2=0$］

同様に $(x^2+y^2)^2-2x^2-y^2=0$ の計算式を描画する。先ほどの $x^2-y^2-(x^2+y^2)^2=0$ と同じ様にプログラムを作成する。そうすると次のような図 14.が得られる。

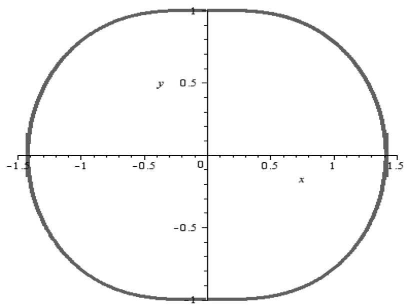

図 14. $(x^2+y^2)^2-2x^2-y^2=0$ の実行結果

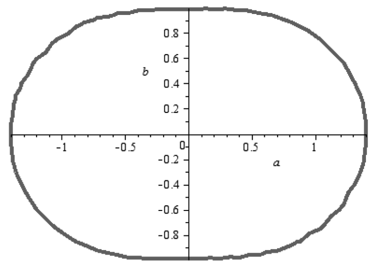

図 15. implicitplot 実行結果

また、implicitplot で上記の数式を描画してみる。どちらの方法が上手く描画できるかを調べる。
図 14.と図 15.は今までの結果と比較してみても、同じような図が描画できている。しかし、良く見てみると多少図 15.の方は線が綺麗な曲線にはなっていない。

3.5　計算方法［$(x^2+y^2)^2-4x^2-y^2=0$］

$(x^2+y^2)^2-4x^2-y^2=0$ の計算式を描画する。それらを合わせると次のような図 16.が完成する。

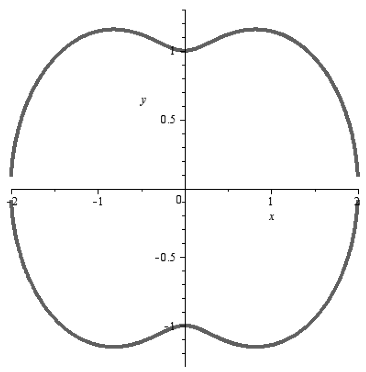

図 16. $(x^2+y^2)^2-4x^2-y^2=0$ の実行結果

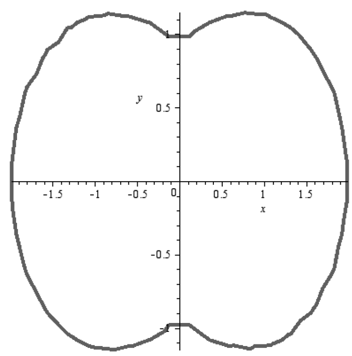

図 17. implicitplot 実行結果

また、implicitplot で上記の数式を描画してみる。どちらの方法が上手く描画できるかを調べる。
この図では比較的今までの結果図からしても、似ていると思う。 $y=1,-1$ の部分で凹んでいる特徴をよ

くとらえている。この図でも、先ほどの描画結果と同様に図 17.のほうが、綺麗な曲線にはなっていない。

3.6　計算例［$x^2y^2(4x^2+y^2)-(x^2-y^2)^2=0$］

同様に $x^2y^2(4x^2+y^2)-(x^2-y^2)^2=0$ の計算式を描画する。

先ほどの $x^2-y^2-(x^2+y^2)^2=0$ と同じ様にプログラムを作成する。そうすると次のような図 18.が完成する。

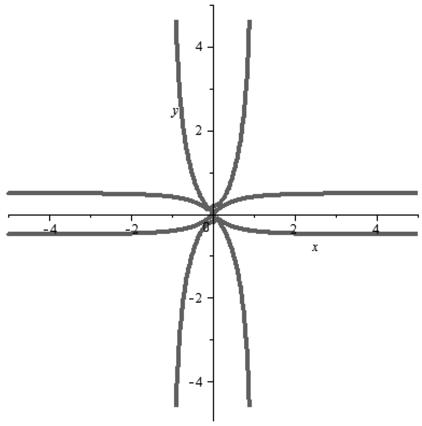

図 18. $x^2y^2(4x^2+y^2)-(x^2-y^2)^2=0$ の実行結果

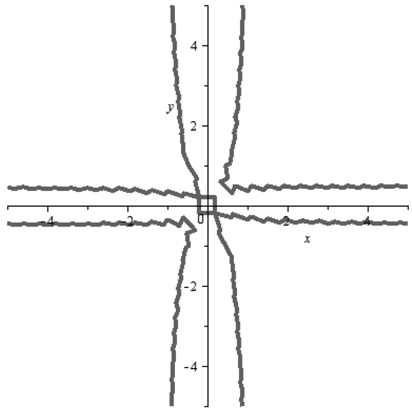

図 19. implicitplot 実行結果

また、implicitplot で上記の数式を描画してみる。どちらの方法が上手く描画できるかを調べる。図 18.と図 19.を比較したら、一目で見てわかるようにこの場合は図 19. implicitplot より図 18.の部分分けプログラムで作成する方法で描画するほうが上手く描けることが分かる。図 19.では $x,y=0$ の辺りで線が折れている状態である。そして線が曲線はなく、波状の線になっていることが分かる。

4　まとめ

代数曲線の描画を微分方程式の解の軌跡を近似的に追う方法について検証してきた。この方法ではグラフが複雑になるにつれて、場合分けが必要である。一方、Maple のグラフ描画関数の一つである implicitplot では、上手く描画できない場合がある。簡単な曲線の場合でも奇麗な曲線にはならない。微分方程式法では、implicitplot を用いた結果より綺麗に描画できている。

図を良く見てもらうと分かるように場合分けし作成した図は、線が途中で途切れている。これは誤差が生じているために生じていると考えられる。この誤差を少なくするためには、範囲を細かく設定をして計算する方法が考えられる。今後は、微分方程式の近似精度を向上することと、適切な場合分けをいかに自動的に行うかが検討課題となる。

参考文献

[1]Amih SAGAN, 長嶋雲兵, 森義仁『自己触媒反応に現れる多根を持つ 2 変数の非線形方程式の簡便な数値解法』,1999

[2]坂井忠次『グラフと追跡』,培風館，昭和 41

[3] 陰関数の導関数, http://www.geisya.or.jp/~mwm48961/kou2/d_implicit.html,2008

[4]CAI,www.sm.u-tokai.ac.jp/~sugita/Classes/Cai/caikai00.pdf,2009

Google Earth による地磁気の可視化 教育・アウトリーチへの利用
Visualization of geomagnetic field with Google Earth: use in education and outreach

* 畠山 唯達
岡山理科大学 情報処理センター
Tadahiro Hatakeyama
Information Processing Center, Okayama University of Science,
Okayama 700-0005, Japan

1 はじめに

3 年ほど前より、Google Earth[1] を用いて地球磁場 (地磁気) を可視化するプロジェクトを立ち上げ取り組んでいる [2][3][4]。Google Earth は身近な地図や地形を 3 次元地球上に投影するだけでなく、「自分の気に入った地点にマークをつける機能」や「時間軸を入れる機能」、「画像等の外部ファイルを取り込んだり HTML へのリンクをする機能」など、さまざまな付加的機能がある。このような機能を用いると、Google Earth を表現の道具としての利用範囲が大きく拡がる。実際、地球科学や地理学の分野では Google Earth をスクリーンとして利用する分野がたくさんある。技術的には、それぞれの分野のデータやデータへのリンクを含む KML(Keyhole Markup Language) ファイルを作成し、それを公開することで、研究者や一般の人が Google Earth の地球上にそれをプロットして理解する・楽しむという方法をとっている。中でも本プロジェクトでは、特に磁力線という地球の外側にも展開する量を Google Earth 上に 3D 的に投影することで、地磁気を専門としない研究者・学生だけでなく、一般の人や小中高校生にも地磁気のイメージを直感的にとらえてもらうことを主眼としている。

こでまでは、地磁気およびその変化変動を表す磁力線についての KML ファイルをを作成し、公開してきた。本稿では、これまでに行ってきた地磁気変動可視化についての実践の取り組みのうち、特に教育・アウトリーチ活動で行ってきたことを紹介し、今後の活動についての展望を示したい。

2 新規に公開した KML ファイル

これまで 2 年間の報告で、(a)2005 年現在の地球磁場、(b) 過去 1 世紀間 (AD 1900-2005) の地磁気変動モデル (IGRF)、(c) 日本における過去 2000 年間の地磁気方位の変化モデル、(d)400 年前〜現在の地球磁場モデル、(e)7000 年前〜現在の地球磁場モデル、(f) 地磁気逆転途中の地球磁場モデル、という地磁気の様子や変動を表す KML を紹介してきた。この 1 年で新規に公開した KML は以下の 1 つであるが、ほかに過去 3000 年の地磁気変動モデル [5] や地磁気逆転を含むダイナモ計算 [6]、新たな日本における古地磁気データモデル [7] などの公開を予定している。

*hatake@center.ous.ac.jp

2.1 地球ダイナモシミュレーションモデル

地球磁場は主に液体金属で構成される外核 (Outer Core) で電磁流体力学的に生成されている。これを解いて地球磁場を再現するためには、流体力学の方程式、電磁気学の式、熱移動の式、構成方程式、物性の式 (状態方程式)、等をまとめて解いて時間発展しなくてはいけない。近年になって、スーパーコンピュータを利用して、これらの式をすべて考慮して解く 3 次元数値地球ダイナモモデルの構築ができるようになってきた (たとえば [8], [9])。

今回は時間発展のガウス係数を得ることができたモデルについて、約 8 千年分の地磁気の時間発展を KML ファイルにした。櫻庭ら [10] によるこのダイナモモデルでは、軸対称双極子成分 (g_1^0) が非常に強く安定した双極性地球磁場が生成されている。また、この時間中に地磁気逆転などの大きな変化はおきていない。昨年度 KML ファイルを公開した、実際の古地磁気データ・考古地磁気データから求められた過去 7000 年の地磁気変動モデル [11][12] と比較しても、非常に安定したものであることがわかる。

ここでは、このダイナモシミュレーションモデルをもとに地球磁場の磁力線と伏角、偏角等観測成分を計算し、KML ファイルを作成した。(図 **1**)

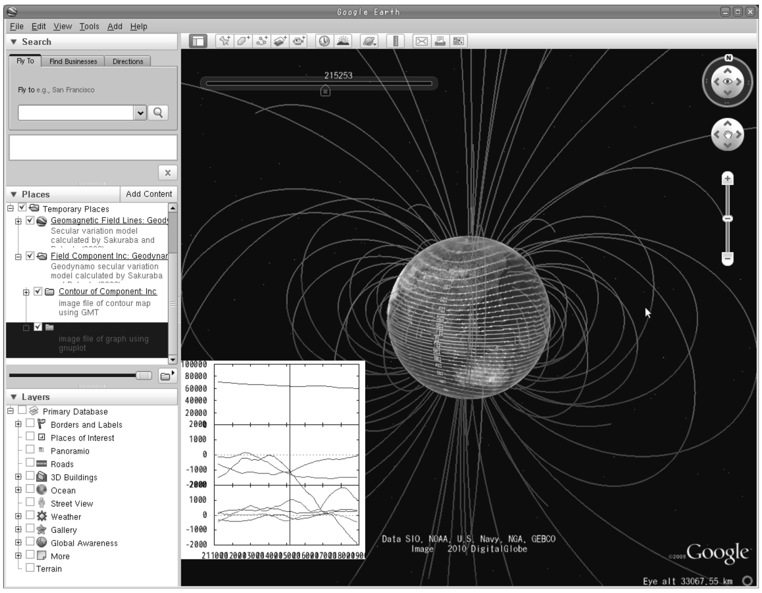

図 1: 地球ダイナモのシミュレーション結果 (Sakuraba and Roberts, 2009) における、磁力線と地表での磁場方位成分。この図では、磁力線と伏角 (I) を同時に表示している。左下の窓には、ガウス係数の低次の項 $(\ell = 1,2)$ の変化を示している。

3 地磁気の教育・アウトリーチへの利用

これまで開発した各種 KML ファイルをただホームページに置くだけでは、地磁気科学の普及にはつながらない。そこで、開発と並行して下記のような活動をしているので紹介したい。

3.1 mini Dagik Earth の作成

Dagik Earth とは、京都大学が中心として進めている Dagik Project で考えられた Google Earth の画面を半球面に投影して手軽に地球らしく見せる仕組みである [13]。半球のサイズは多きものでは数メートルほどあるが、ここでは mini として半径 10cm 程度の半球を貼り付けたスクリーンを作成する。利点としては折り畳みができるようにして、手軽にデモンストレーションができることである (図 2)。投影には

LED を光源とするミニプロジェクタを用い、ノート PC の外部出力に接続する。また、ノート PC 本体の画面では前回の報告 [3] で紹介した、立体視メガネの動画を繰り返し上映することで、2 種類のデモを同時に行うことができる。(図 **2**)

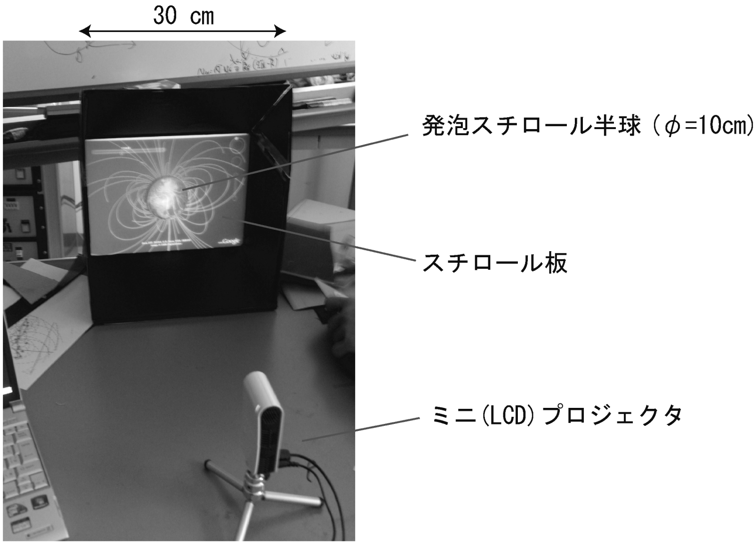

図 2: mini Dagik Earth mini Dagik Earth を用いた投影の例。

3.2　教育用解説ホームページの作成

2007 年 4 月の公開時から、KML へのリンクを置いてごく簡単な使い方の解説を載せたホームページを公開していた。しかし、地磁気そのものやその特徴は学校でほとんど教わらず、あまり知られていない事象であるため、ホームページにおける丁寧な解説が必須であることがわかった (Wikipedia にも地磁気関係用語の丁寧な解説が足りない)。そこで、本年からホームページをリニューアルし、デザインから見直して解説を充実していくことにした。現在までに、地磁気、古地磁気や地磁気成因、変動などの解説をし、関連する KML ファイルや使用法動画の動画を一緒に紹介している。(図 **3**)

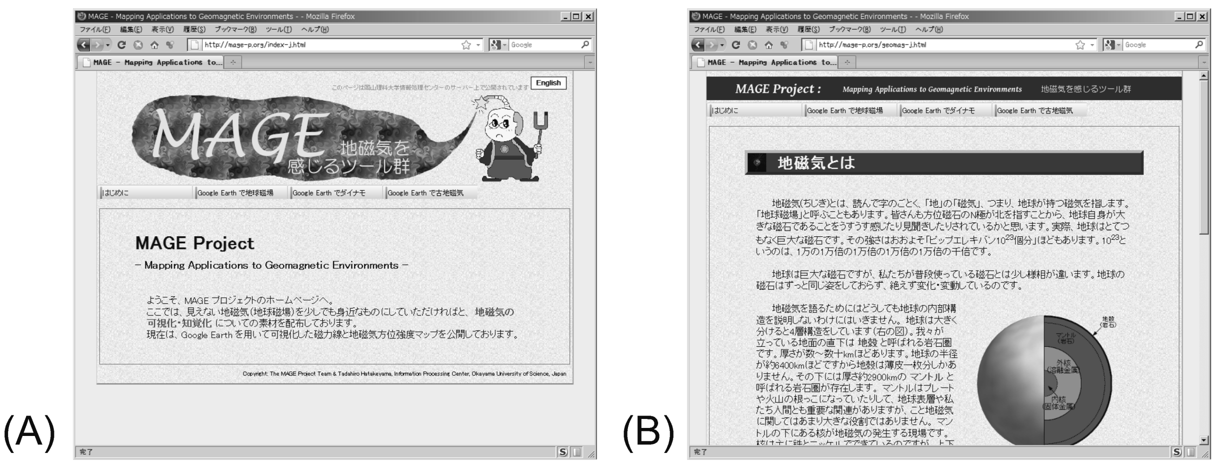

図 3: 新しいホームページ [4] の表紙 **(A)** と地磁気に関する解説のページ **(B)**。

3.3　教育・アウトリーチ活動での使用事例

「地磁気」についての教材として、これまで作成した KML ファイルや mini Dagik Earth, ホームページを活用して数回の講義・講演を行った。2010 年度以降もチャンスがあれば、各所で続けていくつもりである。

- 2009 年 9 月 「古地磁気・岩石磁気夏の学校」にて分野の学生 (学部学生、大学院生) 向けに紹介と実演 (@長野県伊那市)
- 2009 年 10 月 学内講義「地球型惑星の歴史と物質科学」にて画像を紹介
- 2009 年 12 月 (独) 産業技術総合研究所で講演と実演、可視化について多分野の研究者と意見交換
- 2010 年 2 月 (独) 科学技術振興機構「サイエンスパートナーシッププログラム」にて岡山県内の高校生十数名に対する講演で紹介 (@岡山理大)
- 2010 年 2 月 上の一環で、岡山県内高校教員向け講演で紹介と mini Dagik Earth 実演 (@岡山朝日高)
- 他に学会・研究集会での発表が数回ある

また、他の複数の大学で、学部学生に対する講義 (教養・専門) で使用しているとのご報告を頂いている。国内外の小〜大学教育関係者にもさらに宣伝していく必要がある。

4　まとめと今後

Google Earth を用いた地磁気の 3 次元可視化と教育への活用について報告した。利用者からのフィードバックを頂いて、より一層の改良や開発を続ける必要がある。

特に今後すべきことはいくつかある。1 つは映像の 3 次元化である。これまでも既存の立体視の方法を使った 3D 感の表現などをしてきた。昨今巷では 3D テレビが普及の兆しを見せているほか、携帯ゲーム機にも 3D 機能が搭載される可能性などが報じられている。磁力線は 3 次元化して見ることで理解が深まるものだと考えているので、新技術を使った表現法についても開発をしていきたい。また、教育用のソフトウェアとしての親切がまだまだ足りない (KML ファイル内でのコメントがすべて英文、etc.)。小学生にもわかりやすい各種説明を増やしていく必要がある。

2 節で挙げたような新しい KML ファイルの作成も行いたい。また、そろそろ IGRF も次回のものが発表されるのでそちらの更新もしたいと考えている。昨年の報告書で予定していると記述した事項でまだできていないものもあるので、それらについても今後実行していきたいと考えている。

謝辞

Google Earth は Google Inc. の登録標章です。本稿で紹介する KML ファイルを含め、Google Earth の画像等をホームページや印刷物に使うときは、Google 社の許諾をお取り下さい。Dagik Project の皆様には Dagik Earth の使い方をはじめ、ご助言等いただき、大変感謝しております。

References and Notes

[1] Google Earth
`http://earth.google.com/`

[2] 畠山 唯達, Google Earth を利用した地磁気・古地磁気・岩石磁気データの可視化, 岡山理科大学情報処理センター研究報告, **29,** 31–38, 2008.

[3] 畠山 唯達, Google Earth を利用した地磁気変動モデルの可視化, 岡山理科大学情報処理センター研究報告, **30,** 19–26, 2009.

[4] MAGE (Mapping Applications to Geomagnetic Environments)
`http://mage-p.org/`

[5] Korte, M., F. Donadoni and C. G. Constable, Geomagnetic field for 0.3 ka: 2. A new series of time-varying global models, *Geochem. Geophys. Geosys.*, **10**, Q06008, doi:10.1029/2008GC002297, 2009.

[6] Glatzmaier, G. A., R. S. Coe, L. Hongre and P. H. Roberts, The role of the Earth's mantle in controlling the frequency of geomagnetic reversals, *Nature*, **401**, 885–890, 1999.

[7] 山本真央, 日本における過去 2000 年間の地磁気永年変化曲線の復元, 岡山理科大学総合情報学部生物地球システム学科卒業論文, pp, 2010.

[8] Glatzmaier, G. A., and P. H. Roberts, A three-dimensional self-consistent computer simulation of a geomagnetic field reversal, *Nature*, **377**, 203–209, 1995.

[9] Kageyama, A., T. Sato and the Complexity Simulation Group, Computer simulation of a magnetohydrodynamic dynamo. II, *Phys. Plasmas.*, **2**, 1421–1431, 1995.

[10] Sakuraba, A. and P. H. Roberts, Generation of a strong magnetic field using uniform heat flux at the surface of the core, *Nature Geoscience*, DOI: 10.1038/NGEO643, 2009.

[11] Korte M. and C. G. Constable, Continuous geomagnetic field models for the past 7 millennia: 2. CALS7K., *Geochem., Geophys., Geosys.*, **6**, Q02H16:doi:10.1029/2004GC000801, 2005.

[12] `http://mage-p.org/gmf-7ky-j.html`

[13] Dagik (DAta-showcase system for Geoscience In Kml) in Kyoto University
`http://dagik.org/earth/`

Windows 7 上での MPI 環境の構築と fortran 90 からの利用

知能機械工学科 荒木圭典

現在、スーパーコンピュータを利用した大規模な科学計算では、並列計算の API を用いたプログラミングが必須となっている。そのため並列計算のプログラムを作成、デバッグするには並列計算のプログラミング環境が必要となる。その一方で、個人利用用途のコンピュータ(パソコン, 以下 PC)のプロセッサにもマルチコアなものが一般に普及しており、複数台の PC を利用しなくても並列計算プログラムの開発、動作の検証、デバッグ等がある程度できるようになっている。本稿では並列計算の標準的な API の一つである MPI と fortran 90 のコンパイラを無料で Windows 上にインストールし、利用する方法について説明する。内容は 2010 年 11 月の段階の情報であり、筆者の手元の環境は Core i7 640LM, 2GB RAM, Windows 7 Professional (32bit) である。

1. MPI のインストール

現在、Windows 上の MPI 環境でフリーで利用できるものに MPICH2 がある。MPI とは MPI フォーラムによって標準化された規格の名称であり、MPICH2 とはアルゴンヌ国立研究所によって開発された MPI の実装の一つであり、ウェブからフリーで手に入れることができる。

MPICH2 のダウンロードサイトのアドレスは以下のとおりである:

http://www.mcs.anl.gov/research/projects/mpich2/downloads/index.php?s=downloads

ここで 32bit 版の MPI 環境の自動インストーラである mpich2-1.3.1-win-ia32.msi をダウンロードする。インストーラを実行し、画面の指示にしたがってインストールするとよい。途中で何か所か質問のフォームが現れるが、フォーム内のデフォルト値でインストールをすればよい。インストールの最中にファイアウォールのアクセス権の変更の許可を求める画面が現れるが、これは MPICH2 の利用する TCP ポートを開ける作業なので、アクセスの許可をすればよい。MPICH2 はデフォルト値のインストールではディレクトリ C:¥Program Files¥MPICH2 にインストールされる。インストールの詳細については文献[1]が詳しいのでそちらを参照されたい。

2. fortran のインストール

プログラミング言語 fortran は主に科学計算をするためのものであり、多くのスーパーコンピュータにおいて標準のプログラミング言語として利用されている。これに対し C 言語はメモリ利用に関して、動的でフレキシブルなコードを書くことができる言語であるが、逆にこのことが、プログラムを実行する際に必要とするメモリの量を事前に予測しづらくしており、コンパイラの最適化には不向きな面があると言える。大規模な科学計算では fortran の利用を考えるべきである。

現在、fortran 言語は fortran 90 と呼ばれる仕様(もしくはその後継)を利用するのが標準的であるが、数値計算ライブラリの多くはレガシーな fortran 77 のコードに組み込んでも利用ができるように、fortran 77 の固定形式のコードとインターフェースを持たせてあるものが多い。

さて現在、MPICH2 が利用できる fortran 90 のフリーのコンパイラとして g95 Project によってメンテナンスされている g95 がある。g95 は gcc をベースとした fortran コンパイラの一つであるが、gcc project の一部ではなく、それとは独立して開発が進められている。

g95 のダウンロードサイトにある Windows 用の自動インストーラの実行ファイルの URL は

http://ftp.g95.org/g95-MinGW-41.exe

である。インストーラを実行し、画面の指示にしたがってインストールするとよい。途中で何か所か質問のフォームが現れるが、フォーム内のデフォルト値でインストールをすればよい。g95 はデフォルト値のインストールではディレクトリ C:¥MinGW にインストールされる。

3.　MPI を用いた fortran コードのサンプル

MPI を用いた並列計算のコードの基本的な考えは、各プロセッサで同じ宣言文と命令文が独立して並走することである。並列計算の API に OpenMP があるが、これは一つのソースコードの宣言文と命令文を複数のプロセッサで併用して走らせるという考えに基づいている。ここでは HELLO という文字列を並列して実行、表示させるコード `testMpi.f` を紹介する(ファイルの拡張子 `.f` は fortran 90 のソースを固定形式で作成するときの標準的な拡張子の一つであり、固定形式は fortran 77 とソースレベルで互換性を持たせることもできる形式である)。

```
      INCLUDE 'mpif.h'
      CALL MPI_Init(ierr)
      CALL MPI_Comm_size(MPI_COMM_WORLD,nPE,ierr)
      CALL MPI_Comm_rank(MPI_COMM_WORLD,nID,ierr)
      PRINT *,'HELLO',nPE,nID
      CALL MPI_Finalize(ierr)
      END
```

このソースコードを次のコンパイラオプションをつけてコンパイルすると実行ファイル a.exe が作成される。

```
g95 testMPI.f -I"C:¥Program FIles¥MPICH2¥include" -L"C:¥Program
FIles¥MPICH2¥lib" -lfmpich2g
```

ここで `-I` はインクルードパスの指定、`-L` はライブラリのパスの指定、最後の `-l` オプションは事項ファイルにリンクするライブラリの名前である。

4.　並列計算の実行

MPI インターフェースを組み込んだ実行ファイル a.exe を実行するには、MPI のローダである mpiexec.exe をコマンドラインから実行すればよい。実際には a.exe を作成したディレクトリでコマンドプロンプトを起動して次の命令を打ち込めばよい。

```
mpiexec -n 4 a.exe
```

これで自動で並列実行が行われる。

5.　MPI ランタイムの実行権限について

Windows XP 以降の Windows OS では TCP ポートを利用した通信に対するファイアウォールが導入されている。MPICH2 を利用するには、たとえそれがスタンド・アローンの環境での並列化のテストであっても、MPI 通信のための TCP ポートの開放が必要なので、適宜、ポートを開ける必要がある。

さらに Vista 以降の OS では、実行ファイルの実行に際してアクセス権、実行権のチェックが厳しくなっている。MPICH2 の場合、実行時のランタイムとして mpiexec.exe と smpd.exe の 2 種の実行ファイルを動かす必要がある。これらを実行する際には、Windows OS のレベルで「管理者として実行」できるように環境を設定する必要がある。

参考文献

[1] 中野 達也,「MPICH2 による並列計算環境の構築　Windows Vista / 7　編(64bit 環境対応)」, (2010 年 12 月 24 日アクセス)
http://moldb.nihs.go.jp/abinitmp/MPICH2%E7%92%B0%E5%A2%83%E3%81%AE%E6%A7%8B%E7%AF%89%20Windows%20Vista,%207%20%E7%B7%A8%EF%BC%8864bit%E7%92%B0%E5%A2%83%E5%AF%BE%E5%BF%9C%EF%BC%89%2020100819TN-0.pdf

岡山理科大学情報処理センター研究報告　第31号

LMS「MOMOTARO」の移植性向上のための考慮

井上 紀明　大西 荘一　井上善勝　井川真弓　三木啓一郎

岡山理科大学大学院 総合情報研究科 情報科学専攻

1. はじめに [1]

近年，高等教育機関ではICTを活用した教育が盛んになりつつある．ICTを活用した教育支援環境として，CMS（Content Management System）やLMS（Learning Management System）などがあげられる．その中でも，教育活動におけるプラットフォームとしてLMSの導入が重要と考える．

加計グループ大学間で実施されている広域遠隔授業「加計サイバーキャンパス」は岡山理科大学（以下，本学）で独自開発されたLMS「MOMOTARO」で構築されている．加計サイバーキャンパス以外でもMOMOTAROの導入は着実に増加しており，ユーザの戦略・制度・組織の変更に対応できる高い移植性が要求されてきている．本稿では，LMSの開発・運用・保守を経て培ったノウハウを元に，LMSにおいて有用な移植性について，仕様の柔軟性とプログラムの柔軟性の二つの観点から考察した．

2. 加計サイバーキャンパス

サイバーキャンパスとは，インターネット上に構築されたキャンパスである．

本学では，平成14年度から単位認定を伴うインターネットを利用した遠隔授業を実施している．平成17年度から高大連携，大大連携の取り組みをより組織的に行なっていくため，新しい教育の取り組みとして加計サイバーキャンパスを開始した．平成22年度では，5高校，5大学が加計サイバーキャンパスを利用している．平成21年度の受講生の延べ人数は5000名を超えている．

図1は加計サイバーキャンパスの学習管理モデルである．

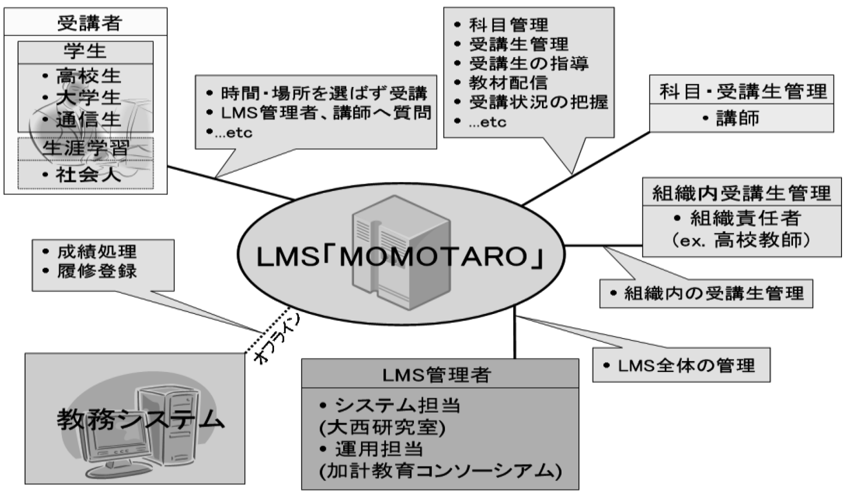

図 1　加計サイバーキャンパスの学習管理モデル[2]

3. LMS「MOMOTARO」

3.1 LMS「MOMOTARO」とは

MOMOTAROの開発は平成15年度から行われ，現在はMOMOTARO Ver9となっている．開発言語は主にPHPを用いている．MOMOTAROの主な機能は以下の通りである．

・VOD・教材コンテンツの配信　・受講生管理　・科目管理　・出欠管理　・組織管理
・パスワードリマインダリ機能　・多重ログイン防止機能　・レポート提出機能
・テスト出題採点機能　・アンケート集計機能　・ライブ授業サポート機能
・アクセスログ解析　・コミュニケーション機能(メール・掲示板・チャット)
・携帯電話・携帯端末対応機能

表１にMOMOTAROの構成，表２にファイル構成を示す.

表１　MOMOTAROの構成

項目	スペック
Webサーバ	Apache 2系列
スクリプト言語	PHP 5系列
	JavaScript
データベース	MySQL 5系列
メールサーバ	Postfix

表２　ファイル構成

種類	ファイル数	実ステップ
PHP	118	33472
JS	1	1085
合計	119	34557

MOMOTAROは加計教育コンソーシアムが運営している「加計サイバーキャンパス」，本学の情報処理センターが運営している「岡山理科大学学習管理システム」，吉備国際大学が運営している「メディア授業」，文部科学省の戦略的大学連携支援事業（以下，戦略GP）に採択された「科学Tryアングル岡山」が運営している「KTOサイバースクール」，同じく戦略GPに採択された「岡山オルガノン」が運営している「まなびオルガノン」の各e-LearningサイトのLMSとして利用されている.

３．２　e-Learningサイトごとの特色

e-Learningサイト[3] [4] [5] [6] [7]によってLMSに要求される機能は一様ではない. そこで，現在稼動している５台のサイトの特色を組織・受講者・内容の違いから述べる.

表3　e-Learningサイトごとの違い

e-Learningサイト	組織	受講者	内容
加計サイバーキャンパス	加計学園5大学 教育協定5高校	連携校の学生 高校生	大大・高大連携による単位互換
岡山理科大学 学習管理システム	1大学内での運用	大学内の学生	学内の講義補助
吉備国際大学 メディア授業	2大学での運用	大学内の通学学生・通信学生	大大連携による単位互換
KTOサイバースクール	岡山県内3大学 1高専	連携校の学生一般市民	科学の普及を目的とした学習
まなびオルガノン	岡山県内15大学	連携校の学生	15大学連携科目の単位互換

４．仕様の柔軟性

MOMOTAROは加計サイバーキャンパスでの運用のみを想定して設計されてきたため，仕様の柔軟性が乏しい. 一例として，学習者からのお問い合わせの際などに使用されるメールアドレスを挙

げる．そのメールアドレスは10個のPHPファイルに記述されており，運営が変わるなどしてメールアドレスの変更が起きた際，10個のファイルすべてを修正しなければならない．これらの問題は，開発者の負担増加やバグの潜在など，様々な弊害を招く可能性がある．そのため，LMSを移植する際に，予め変更が想定される機能については，システム上で変更できるようにするほうが効率的であると考えた．そこで，変更の要望が多い機能については，システム上で設定を行えるようにした．

以下に設定の例を示す．

(1)レポートの提出可能な拡張子の変更

組織あるいは講師によって，提出させたい拡張子は様々である．例えば，ある組織ではPPTファイルの提出は必要ないが，別の組織ではプレゼンテーションの練習としてPPTファイルを提出させたいことがある．しかし，すべての拡張子を許容してしまっては，講師の意図しないファイルが提出される可能性があるため，制限が必要である．

(2)アップロードサイズの最大値設定

アップロードする教材やレポートのファイルサイズの上限を設定することができる．MOMOTAROサーバのメモリ容量により，扱うファイルサイズの最大値は必要分を設定するほうが望ましい．

(3)メールアドレスの変更

お問い合わせの際などに使用されるヘルプデスクのメールアドレスを変更することができる．組織によって，使用するメールアドレスは異なる．同一組織であっても，メールアドレスの変更が起きる可能性がある．そのため，運用中でも容易に変更ができるようにしている．

(4)ヘッダの変更

ヘッダの変更をシステム上で行えるようにすることで，外観を大幅に変更しない場合は，画像の差し替えだけで移植することができる．

(5)IDのマニュアル・自動登録切り替え

IDの生成をMOMOTAROが登録時に自動で行うか，マニュアルで登録するかの切り替えである．
複数の組織が連携する場合は自動設定に，単独の場合はマニュアル設定にすることが想定される．

(6)LMSの初期化

データベースの初期化を行うことができる．初期データをデータベースに直接マニュアル入力せずに，自動的に行える．

5. プログラムの柔軟性

様々なニーズに対応しようとすると，仕様としての柔軟性だけでは補えない場合が多々ある．その場合は，ソースコードを変更する必要があるが，現状のMOMOTAROでは，ソースコードの改変に多大な労力を費やしてしまう．ソースプログラムの改変が容易なアーキテクチャにする必要がある．

5.1 アーキテクチャ

アーキテクチャとはシステム全体の設計思想や構造のことである．多層アーキテクチャやMVCアーキテクチャなど，様々な概念が存在する．

PHPは非常に簡単にWebアプリケーションを作成することができるが，その反面，保守性や再利用性の低い「ページ志向の設計」になりがちである．このことはMOMOTAROにおいても当てはまる．

５．２　MOMOTAROのアーキテクチャ

MOMOTAROは多数の開発者が8年間にわたり，その時々の要求仕様の実現を最優先に開発してきた．そのため，プレゼンテーション，データベースアクセス，ビジネスロジック，イベント処理を一塊のコードとして記述するという，モノリシックなアーキテクチャとなっており，重複・類似した無駄なコードが多数存在している．

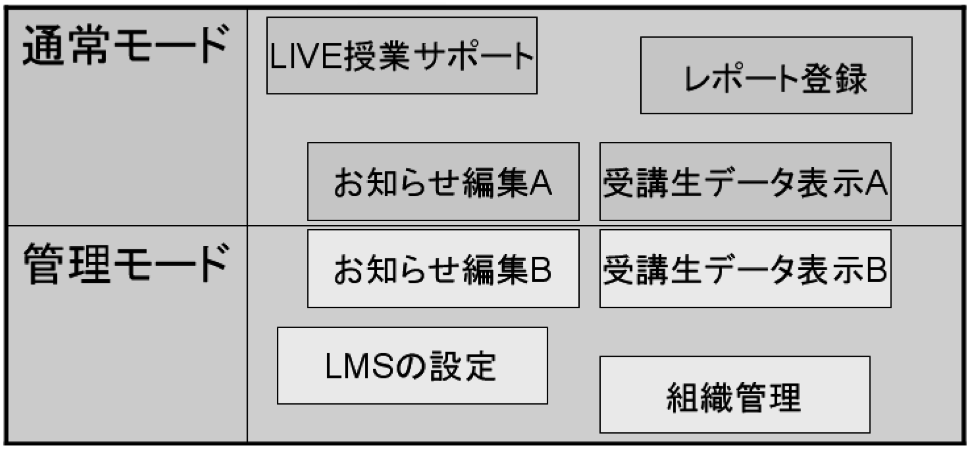

図2．機能別ファイル構成

図２は機能別ファイル構成の一部である．MOMOTAROには受講生が講義を受講するための通常モードと，管理者のみが操作でき，受講生データ管理や科目管理等の各種管理を効率よく行える管理モードがある．図２で見られるように，同じお知らせ編集機能であっても，通常モードと管理モードで個別にファイルが存在している．この問題により，機能の追加・改変などを行う際に多くのコードを修正する必要があり，システムを深く理解した者でなければ開発・保守を容易に行うことができない．

しかし，より効率的で移植性の高いシステムを実現するためには現在のアーキテクチャでは不十分である．そこで，MOMOTAROの現状を踏まえ，本稿ではMOMOTAROの移植性を高めるMVCアーキテクチャの提案を行う．

５．３　MVCアーキテクチャ [8]

MVCアーキテクチャとは，プログラムをModel・View・Controllerという３種類のコンポーネントに分割して設計・実装する技法である．それぞれの役割を以下に示す．

・Model

アプリケーションが扱う領域のデータとビジネスロジックを保持するためのコンポーネントである．扱うデータは，データベースなどの外部に格納される永続的なものが多いが，ユーザとの少しのやり取りの間しか保持されない一時的なものもある．

ビジネスロジックをModel内に実装することで，データの有効性を損ねるような処理をアプリケーションのほかの部分が行わないように保証できる．

・View

Modelに格納されているデータに基づいて，ユーザインタフェースを生成するコンポーネントである．Viewはデータ入力の様々な方法をユーザに提供するが，View自体が入力データを扱うわけではない．例えば，Webアプリケーションにおいて，HTML文書を生成して動的にデータを表示するためのコードなどにあたる．

・Controller

アプリケーション内の制御役となるコンポーネントである．ユーザからのイベントに対して

応答し，Modelとやり取りをし，適切なViewをユーザに表示する．ModelとViewに変更を引き起こす場合もあるが，直接Webブラウザに描画を行ったり，データベースの操作を行ったりはしない．

図３にMVCアーキテクチャのモデル図を示す．

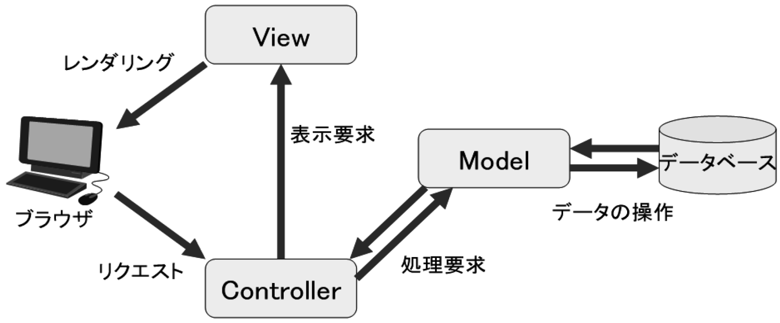

図3　MVCアーキテクチャのモデル図

ブラウザからの生データをControllerが受け取り，ControllerはModelに生データを渡し，Modelは生データの操作を行いControllerにデータを返す．そしてControllerはModelから返ってきたデータをViewに渡し，ViewはControllerから受け取ったデータを元にレンダリングしブラウザにレンダリングされたデータを渡す．

5．4　MVCアーキテクチャの実装

図４はMVCアーキテクチャの簡単な実装例である．

```
<?//フォームから受け取ったお知らせを
  //表示するだけのファイル(news.php)
?>
<form action="<?=$_SERVER["PHP_SELF"]?>" method="post">
<p><input type="text" name="news" /></p>
<p><input type="submit" name="submit" value="送信" /></p>
</form>

<?php
    $news = isset($_POST["news"]) ? $_POST["news"] : NULL;
    if($news){
        $news = htmlspecialchars($news);
        echo "<p>お知らせ：{$news}</p>";
    }else{
        echo "<p>お知らせを入力してください。</p>";
    }
?>
```

```
<?
//Model部分(NewsModel.php)
class NewsModel{
    private $news;
    function newsRegist(){
        if($this->news){
            return "お知らせ:" . $this->news;
        }else{
            return "お知らせを入力してください。";
        }
    }
}
?>
```

```
<?
//View部分(NewsView.html)
?>
<form action="<?=$_SERVER["PHP_SELF"]?>" method="post">
<p><input type="text" name="news" /></p>
<p><input type="submit" name="submit" value="送信" /></p>
</form>
<p><?=htmlspecialchars($news,ENT_QUOTES)?></p>
```

```
<?
//Controller部分(NewsController.php)
require_once("NewsModel.php");
$news = isset($_POST["news"]) ? $_POST["news"] : NULL;
$newsObj = new NewsModel;
$newsObj->news = $news;
$news = $newsObj->newsRegist();
require_once("NewsView.html");
?>
```

図4　MVCアーキテクチャの実装例

左のソースコードがMVCアーキテクチャの適用前，右が適用後である．図４のように，様々な処理が一塊になっているソースコードをModel・View・Controllerの役割に分離し，それぞれを別のファイルとして実装する．ただ機能を分離するだけでなく，Viewは生データを扱ってはならない．またModelは入出力に関するデータを扱ってはならない．大規模なソースコードの場合，ControllerとModelはクラスを用いて実装し，１つの機能ごとに１つのクラスを用意する．

この様な実装をソースコード全体に施していかなければならないため，フレームワークを活用するなどして，実装の効率化を図る必要がある．

表4はMVCをサポートしている代表的なフレームワークである．

表4　フレームワーク

言語	フレームワーク
PHP	ZendFramework CakePHP Simfony
JAVA	Struts
Ruby	Ruby on Rails

5．5 MVCアーキテクチャのメリット［9］

MVCアーキテクチャはソースコードの規模が大きいほどメリットが得られる．MVCアーキテクチャを採用することで，コンポーネント間の依存性が最小限に抑えられ，役割分担が容易になる．ControllerとModelはプログラマ，ViewはWebデザイナーが開発するなどの分業が可能で，効率的にシステム開発を行うことができる．

MOMOTAROの移植性という観点からみると，システムを別の環境に移植する際に，ハードウェア的な差異はControllerの変更で対応できる．システムの外観を変更する際はViewのみの変更で対応でき，ユーザインタフェースの変更も，ViewとControllerの変更までで対応できる．そのため修正箇所が明確であり，効率的に作業を行うことができる．機能の追加や改変をする際にも，モジュールを細かく分離しているので，再利用性が高まり，開発の効率化が望める．また，機能ごとの独立性が高く，他の部分の影響を受けにくいため，連鎖的な不具合が発生し難い．そのため，不具合の原因特定や修正が容易になり保守性が向上する．プログラミングの担当者が短期間で入れ替わるMOMOTAROの開発にとっては理想的であるといえる．

6．おわりに

本稿では，移植性の向上を図るために，「仕様の柔軟性」と「プログラムの柔軟性」という２つの観点から考察した．アーキテクチャについては，現段階ではMVCアーキテクチャを検討中であり，実装には至っていない．仕様の柔軟性については，現在実動している５サイトのニーズの違いを設定でカバーできている．これにより移植作業はかなり効率的に行うことができるようになった．

今後の課題として，仕様の柔軟性という観点からは，今後出てくる仕様変更についても対応していく必要がある．プログラムの柔軟性という観点からは，フレームワークを活用するなどして，MVCアーキテクチャを適用し，プログラムの保守性・可読性・拡張性の向上を目指す．

参考文献

［1］井上紀明，大西荘一，井上善勝，井川真弓，三木啓一郎，
「LMS「MOMOTARO」の移植性向上のための考慮」，
平成22年日本教育情報学会第26回年会 pp.42-45

［2］小池崇，大西荘一，井上紀明，井上善勝，
「学習履歴データの分析によるLMS「MOMOTARO」の機能改善とモバイル対応」，
岡山理科大学情報処理センター研究報告　第30号 pp.1-7

［3］加計サイバーキャンパス　https://cyber.kake-group.jp/

［4］岡山理科大学学習管理システム　https://momotaro.std.ous.ac.jp/

［5］吉備国際大学メディア授業　https://media-tei.jp/

［6］ＫＴＯサイバースクール　https://cyber.kagaku-try-okayama.jp/

［7］まなびオルガノン　https://manabi-organon.jp/

［8］MVC モデル　http://ja.wikipedia.org/wiki/Model_View_Controller

［9］Sam Ruby, David Heinemeier Hansson, Dave Thomas 著　前田修吾 訳
「Rails によるアジャイル Web アプリケーション開発」オーム社；第３版 2009/12