4 Mapleを利用した応用数学教育

京都大学工学研究科材料工学専攻（材料設計）　西谷　滋人

＜概要＞
数式処理システムMapleをもちいて、材料科学コース3回生の演習を数年前から実践している。数式処理システムは数学やプログラミングを専門としない学生が、数学を道具として使える能力を身に付けるには理想的なソフトである。しかし受講生の追跡調査では数式処理ソフトを自由に使いこなす程度まで習熟するには至っていない。数式処理ソフトのとっつきの悪さについて分析し、どのような教授法が望ましいかを考察した。

キーワード：Maple、数式処理システム、計算材料学、巡回セールスマン問題

Applied Mathematics Education Using Maple

Shigeto R. NISHITANI

Using Maple, a Symbolic Computation System or Computer Algebra System, we have an exercise course on material science for the undergraduate students of the third grade. Maple should be the ideal tool of mathematics for the students not majoring mathematics nor programming. The follow-up review, however, revealed that the students who took this course are not applying this software on daily researches. We will explore the reasons of this situation and revise the teaching method.

Keywords: Maple, Material Science, Traveling Salesman

1．知識のユーザーとメーカーの視点

“Any sufficiently advanced technology is indistinguishable from magic.”

有名なSF作家Arthur C. Clarkeの第三法則です。はじめて手持ちのMac上の数式処理ソフトが、複雑な数式を一瞬にして簡単化したのを目のあたりにしたとき、科学がまた一歩魔法に近づいた気がしました。それ以来数式処理ソフトは必須の道具として私のMacに10年以上常駐しています。

MapleというソフトはC&Eの読者がよくご存じのMathematicaと同じ範疇にある数式処理ソフトです。MapleとMathematicaの比較は数理科学11/1998にあります[1]。結論は引き分けというところでしょうか。それほど厳しい使い方をしていない私には、そこにMapleがあったから使っているというのが本音です。事実最初に使いはじめた数式処理ソフトはMathematicaです。さらに、Mapleを使い込んでいるうちにMathematicaのコマンドは忘れてしまったので、MathematicaとMapleの比較ももはやできません。

私が本業としているのは材料工学の計算機シミュレーションです。材料工学という分野になじみがないと、以下の文章での学生の状況がわかりにくいので、すこし詳しく触れておきます。材料はFig. 1 にあるとおり、物質を電子・原子レベルでとらえる物理・化学と、連続体とみなす機械工学や電子工学との間にあって、組織制御やプロセスを研究する学問分野です。

材料工学の計算機シミュレーションを研究している同業者は、鉄鋼・素材メーカーや電気メーカーの部品開発部門

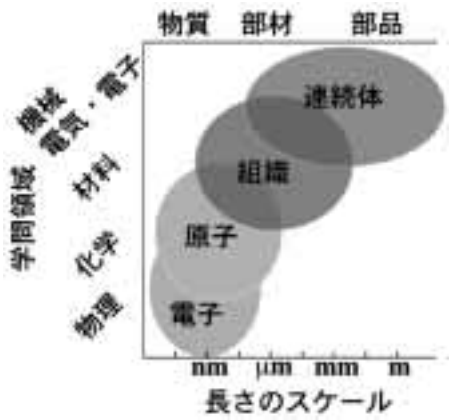


Fig.1　材料科学の研究分野

で物質設計、材料開発に計算機を利用する研究をおこなっています。部品あるいは部材関連企業間には明らかにメーカーとユーザーという系列があって、だいたいメーカーがユーザーにたたかれるという構図ができ上がっています。知識にもメーカーとユーザーという区別があり、われわれ材料研究者は数学、数値計算のユーザーです。これ以外にも学習すべき知識が多いため、それらの科目の教育に十分な時間を割けません。ところが、知識の系列では決してユーザー優位という構図はできておらず、数学を使いこなすレベルに到達するまでに相当の時間がかかります。

Mapleは、ユーザー側の視点で作られた数少ない数学の道具の一つです。Mapleを初めとする数式処理ソフトは、すくなくとも材料研究のように数学やプログラミングを専門としない学生を育てるうえに最適な、日常の研究を

Contact to : e-mail:nishitani@mtl.kyoto-u.ac.jp, http://www.mtl.kyoto-u.ac.jp/users/bob

進めるうえでは必須の道具であると私は信じています。数式処理ソフトの俯瞰的な話、あるいは一般論を構築する能力も暇もないので、材料学あるいは応用物理の理論計算の道具としてMapleを使った授業内容を紹介し、現場で得た経験から分析を試みます。

2. 物理工学演習でやっていること

2.1 演習の概要

Mapleを材料科学コースの演習でどのように使っているかを初めに概説します。材料科学コースの3回生55名を対象に、選択必修科目の物理工学演習の前期の半分(約7コマ)をMapleを使った演習にあてています。彼らは数学、物理の基礎科目、および計算機数学を履修してきています。

Maple導入時(96年)の問題点は

大学の講義は座学で十分

自分の手で計算してグラフを書いて……という演習を大学に入ってからはほとんどしていません。これは学生側だけの問題でなく、教える側も式の導出や数値を入れた計算は時間ばかりとるので、単純なモデルで説明するか、自習に任せてしまって、概要をなぞるだけで終ってしまいがちです。

数学恐怖症

大学入学後の範囲となる固有値、複素関数、微分方程式などの単語が出てきただけで思考停止に落ちいる学生がいます。

プログラミング恐怖症あるいはC言語至上主義

計算機数学でFortranを使ったプログラミングの初歩をかじっていますが、自分でプログラムを書くというレベルまで到達している学生はまれで、多くの学生はかえってプログラミング恐怖症を植え付けられています。もう一方の極端には、C言語(今はC++, Javaなどでしょうか)などのよりすぐれた(と耳学問で聞いている)プログラミング言語しか使いたくないと公言する学生もいます。

そこで物理工学演習では

材料工学3回生で使う実際の計算を示し、教科書の記述から問題を組み立てそれを解く手法の習得

をめざしました。このような目標に対して、Mapleを数学・数値計算を受け持ってくれる便利な道具として導入しました。

材料あるいは応用物理の領域で計算問題を解くときには、本来その基礎となる物理・数学・プログラミングの知識が不可欠です。入力を変えたり、結果を加工したり、いろいろと問題のまわりで遊ぶことによって問題を理解し解いていくという作業がつづきます。このような段階を一手に引き受けてくれる専用ソフトもいくつかでています。MapleはFig.2に示した、数学・プログラミングおよび専用ソフトの領域を受け持ってくれる強力な道具となります。

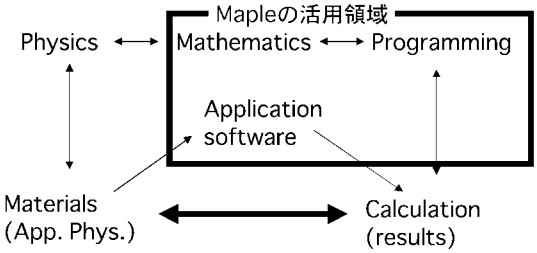


Fig.2　Mapleの活用領域

演習形態は

1）教科書の抜き刷りを配付、
2）必要な情報をとりだし、問題を解く、
3）Mapleの基本操作と理論の基礎を教え、あとは自習、
4）TA二人と講師とで机間巡視で50名程度を指導、
5）レポート提出を課しました。

レポート課題とその内容をTable 1に示します。テキストは参考文献［2］にあります。

Table 1　レポート課題とその内容

量子トンネル効果（式の変形）
熱膨張係数の導出（複雑な関数の近似と積分）
拡散方程式の解(微分方程式)
2原子分子のHamiltonianの解（線形代数）
組成自由エネルギー曲線（連立方程式の数値解）
ノイズフィルター（高速フーリエ変換)
カーブフィッティング（非線形最小二乗法）
巡回セールスマン問題(simulated annealing法)

演習で取り上げているトピックをいくつか紹介しながら、どのようなレベルを目指しているかを示したいと思います。以下に示した程度のスクリプトはいまのところ、機種(Windows, Mac, Linux, Unix)やバージョン(MapleVR4-Maple8)に依存せずに実行可能です。現在、私はMaple7をMacOS9上で、講義ではWindows2000上で使っています。現在の最新バージョンはMaple 8です。参考のため、Mapleの出入力画面（Mac上のMaple 7で作成）の例をカラー図面で本文カラーページに掲載しています。

2.2　Mandelbrot(マンデルブロー)集合

初めに、複雑な問題に対してMapleがどれだけ直観的かを示す例として、かの幻想的で美しいMandelbrot集合の描画を取り上げます。コンピュータアートの草分け的存在です。フラクタルの難しい内容はおいといて[3]、プログラムの参考書をひもとくと[4]

Mandelbrot集合　Mandelbrot set

（中略）

計算機では、ある領域の点(x,y) について、

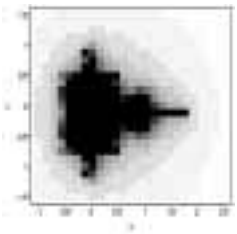
```
z <-- x + i y; count <-- M;
while ( |z| <= 4) and ( count>0) do begin
```

```
z <-- z^2 - (x + i Y); count <-- count -1
end;
点(x,y) にcount で決まる色(count=0 なら黒)を付ける;
```

とする。黒い部分がMandelbrot集合で、それ以外の色は、その点とMandelbrot集合との“近さ”を表す光背である。となっています。この後、プログラムコードが続き、どこかにあるGraphics表示用の謎のルーチンを呼ぶように指示してあります。

これをMapleで実現しようとすると、

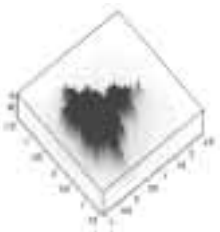
```
mandel:=proc(x,y)
  z0:=x+y*I;
    z:=0;
  count:=20;
  while (evalf(abs(z))<4.0 and count>0) do
    z:=z^2-z0;
    count:=count-1;
  od;
  count;
end:
with(plots):
densityplot('mandel(x,y)',x=-1..2.5,y=-1.5..1.5);
```



となります。どうです？！　読み飛ばした方、もう一度、数学的な記述と見比べてください。上のMapleスクリプトでは、先ずmandel(x,y)という関数をアルゴリズムに添って定義して、次にx=-1..2.5,y=-1.5..1.5の範囲でdensityplotさせています。直観的に分かる数学的な記述そのままで、しかもわずか数分で目的とするMandelbrot集合を実際に描くという作業が終了してしまいます。

もう少し見た目にこだわる場合には、

```
with(plots):
plot3d(mandel,-1..2.5,-
1.5..1.5,axes=none,shading=zgrayscale,
style=patchnogrid,grid=[200,200],orientation
=[-50,-170]]);
```



などととします。カラーshading=zhueだともっときれいです。（カラー図は本文カラーページにあります）

2.3　どろくさい例

先程の例はきれいな見本です。実際の演習ではもっとどろくさい例を取り上げます。これはMapleをはじめとする数式処理ソフトの習得にあたって初心者がおちいる共通した間違いを回避するためです。これをまとめると

鉄則 0　restart をかける：続けて入力すると前の入力が生きている。違う問題へ移るときやもう一度入力をし直すときにはrestartを入力して初期状態からはじめる。

鉄則 1　出力してみる：多くのテキストではページ数の関係で出力を抑止しているが、初心者が問題を解いていく段階ではデータやグラフをできるだけ多く出力する。

鉄則 2　関数に値を代入してみる：数値が返ってくるべき時に変数があればどこかで入力をミスっている。plotでPlotting error,empty plot が出た場合にチェック。

鉄則 3　内側から順に入力する：長い入力は内側の関数から順に何をしているか確認しながら打ち込む。

演習で実際に取り上げている次のような積分

$$\int_{-\infty}^{\infty} xe^{(-\beta cx^2)}(1+\beta gx^3)dx$$

を例に、上に掲げた鉄則の具体的な状況を説明します。この解答をテキスト [2] では

```
f1:=unapply(x*exp(-beta*c*x^2)*(1+beta*g*x^3),x);
```

$$f_1 := x \to xe^{(-\beta cx^2)}(1+\beta gx^3)$$

```
assume(beta*c>0);
limit(int(f1(x),x=-alpha..alpha),alpha=infinity);
```

$$\frac{3}{4}\frac{\sqrt{\pi}g}{\beta\tilde{}c\tilde{}^{2}\sqrt{\beta\tilde{}c\tilde{}}}$$

と書いてます。しかし、これはあらかじめ以下に示すような、いくつかの予備計算を裏でした後の結果を短く表記しているだけです。先ず鉄則0にしたがってrestartをかけ、関数を定義します。

```
restart;
  f1:=unapply(x*exp(-beta*c*x^2)*(1+beta*g*x^3),x);
```

$$f1 := x \to xe^{(-\beta cx^2)}(1+\beta gx^3)$$

次には鉄則 1 にしたがって積分する前にどのような関数かプロットしてみます。

そのまま

```
  plot(f1(x),x=-10..10);
```

とすると

```
Plotting error, empty plot
```

と怒られます。これは鉄則２にあるとおり、

```
f1(10);
10*exp(-100*beta*c)*(1+1000*beta*g)
```

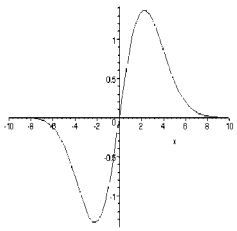
で、beta,c,gなどのパラメータの値が入っていないためとわかります。

適当に

```
c:=1; g:=0.01; beta:=0.1;
```

として、変数に値を代入し、再度プロットを試みると

```
plot(f1(x),x=-10..10);
```



とグラフを書いてくれます。これから-∞..∞の積分がなんらかの値を取ることがわかります。さらに鉄則３にしたがって、式を頭から打ち込むのではなく内側からみていきます。これは問題を解いていく時に思考が必ずたどるであろう順番に相当します。先ず変数に入れた数値をクリアします。

```
c:='c'; g:='g'; beta:='beta';
```

不定積分でこの関数が積分できることを確認し、

```
int(f1(x),x);
```

$$-\frac{1}{2}\frac{1}{\beta c\mathrm{e}^{(\beta c x^2)}}+\beta g\left(-\frac{1}{2}\frac{x^3\mathrm{e}^{(-\beta c x^2)}}{\beta c}+\frac{\frac{3}{2}\left(-\frac{1}{2}\frac{x\mathrm{e}^{(-\beta c x^2)}}{\beta c}+\frac{1}{4}\frac{\sqrt{\pi}\,\mathrm{erf}(\sqrt{\beta c}x)}{\beta c\sqrt{\beta c}}\right)}{\beta c}\right)$$

次にx=-alpha..alphaの定積分を実行してみます。

```
int(f1(x),x=-alpha..alpha);
```

$$-\frac{1}{4}\frac{g\left(4\alpha^3\mathrm{e}^{(-\beta c\alpha^2)}\beta c\sqrt{\beta c}+6\alpha\mathrm{e}^{(-\beta c\alpha^2)}\sqrt{\beta c}-3\sqrt{\pi}\,\mathrm{erf}(\sqrt{\beta c}\,\alpha)\right)}{\beta c^2\sqrt{\beta c}}$$

さらにalpha→∞としてみます

```
limit(int(f1(x),x=-alpha..alpha),alpha=infinity);
```

$$\lim_{\alpha\to\infty}-\frac{1}{4}\frac{g\left(4\alpha^3\mathrm{e}^{(-\beta c\alpha^2)}\beta c\sqrt{\beta c}+6\alpha\mathrm{e}^{(-\beta c\alpha^2)}\sqrt{\beta c}-3\sqrt{\pi}\,\mathrm{erf}(\sqrt{\beta c}\,\alpha)\right)}{\beta c^2\sqrt{\beta c}}$$

ところがこれでは答えを返してくれません。積分した後のそれぞれの項を見るとbeta*c>0を仮定すれば簡単になることが分かります。このような変数の仮定(assume)は

```
assume(beta*c>0);
```

でおこないます。結果として最初に出した解答

```
limit(int(f1(x),x=-alpha..alpha),alpha=infinity);
```

$$\frac{3}{4}\frac{\sqrt{\pi}g}{\beta\tilde{}c\tilde{}^2\sqrt{\beta\tilde{}c\tilde{}}}$$

が導かれるのです。ここで～記号は変数に何らかの仮定を施していることを示しています。最新のMapleでは改良が施されていて、このような複雑な積分も一発で

```
int(f1(x),x=-infinity..infinity);
```

$$\begin{cases}\dfrac{3}{4}\dfrac{g\sqrt{\pi}}{\beta c^2\sqrt{\beta c}} & \mathrm{csgn}(\beta c)=1\\ \infty & otherwise\end{cases}$$

と求まるようになっています。ここではβcが正の場合(csgn(βc)=1)とそれ以外の場合(otherwise)に分けて答えを返しています。しかしこのようなきれいな結果だけを示したのでは、なにかうまくいかないときにお手上げになってしまいます。数式処理ソフトを初心者が習得する時には、実際に紙と鉛筆で解いていく手順をスクリプトに置き換えていくだけであるということをどこかで教える必要があります。

2.4　巡回セールスマン問題

次の例は大域的最適化問題あるいはNP(非決定性的多項式)完全問題としてよく取り上げられる巡回セールスマン問題です。材料科学ではモンテカルロシミュレーションの基礎として重要です。だいぶ込み入った課題ですので、なじみのない方のために以下に詳しく内容を記します。

理論

巡回セールスマン問題とは、ある街から出発していくつかの街を次々とめぐって元の街に戻ってくる最短の経路を求める問題です。訪れる街の数が少ないときにはすべての経路を数え上げればいいのですが、数が増えるとその計算時間は指数関数的に増えてしまうと予想される問題です。このような問題はセールスマンだけでなく、コンピュータのCPUの配置や、都市ガスの配管設計などでも出会う問題です。

対象となる経路の長さの合計を

$$E(\boldsymbol{a})=\sum_{i=1}^{N}\left\|\boldsymbol{r}[\boldsymbol{a}_i]-\boldsymbol{r}[\boldsymbol{a}_{i+1}]\right\|$$

としておきます。ここで$\boldsymbol{a}$はそれぞれの街の順番あるいは配置を示しています。$\boldsymbol{r}$はそれぞれ街の座標で$\|\boldsymbol{r}\|$で距離を求めます。するとこの関数は模式的にFig.3のようになると考えることができます。すべての配置について$E(\boldsymbol{a})$を求めれば、最小値が求まります。あるいは初期の配置を

$\boldsymbol{a}$ = [1,2,3,...,N,1]

として、一定の手順で変更$\delta\boldsymbol{a}$を加え、$E(\boldsymbol{a}+\delta\boldsymbol{a})$が下がった場合にその配置を採用するという方法をとることも出来ます。しかし、この方法は極小値に落ちてしまうことが分かるでしょう。最小値を探すには時として坂を駆

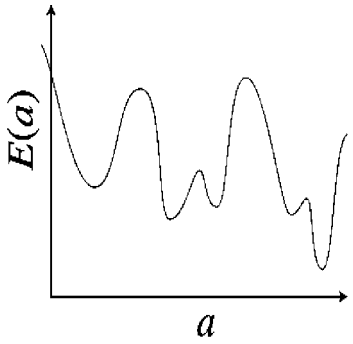


Fig.3　配置による経路の変化

け登る必要があるのです。このような問題の一つの解法としてアニーリング法(simulated annealing)があります。これは格子欠陥を多く含んだ金属を高温へ上げて欠陥を掃き出し柔らかくする熱処理(焼きなまし、annealing)からの類推で名付けられた手法です。

アルゴリズムは以下のとおりです。

1. 配置$\boldsymbol{a}$を仮定し$E(\boldsymbol{a})$を求める。
2. $\boldsymbol{a}$からすこし違った配置$\boldsymbol{a}+\delta\boldsymbol{a}$を作る。
3. $\Delta E = E(\boldsymbol{a}+\delta\boldsymbol{a}) - E(\boldsymbol{a})$を求める。
4. $\Delta E < 0$ なら新たな配置を採用する。
5. $\Delta E > 0$なら新たな配置をexp(-$\Delta E/T$)の確率で受け入れる。
6. 手順2以下を適当な回数繰り返す。

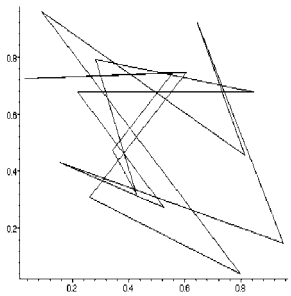
ここでTは温度から類推される制御パラメータで、十分大きい場合はすべての状態が採用されます。一方、Tを下げるにつれて採用される試行が少なくなり、徐々に状態が凍結されていきます。$\boldsymbol{a}+\delta\boldsymbol{a}$の生成方法と$T$の下げ方をうまく取れば最小値に近い状態が確率的に高く出現し、最小値かそれに近い状態へ落ち着くというアルゴリズムです。

課題

街の間の距離をあらかじめ計算して、2次元の配列に入れておき、配置$\boldsymbol{a}$から任意に2都市を取りだして入れ替える試行を$\delta\boldsymbol{a}$として上述のアルゴリズムにしたがって最短経路を探すプログラムを作成してください。

ヒント：16都市の2次元座標をランダムに生成し、セールスマンのルートを表示するスクリプトは以下の通りです。

```
with(plots):with(linalg):
N:=16:
Point:=seq(evalf([rand()/10^12,rand()/10^12]),i=1..N):
a:=[seq(i,i=1..N),1];
Route:=[seq(Point[a[i]],i=1..N),Point[1]]:
PLOT(CURVES(Route));
```



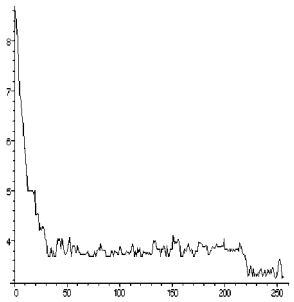
また、2..Nの乱数を生成する関数は次のようにします。

```
Sel_city:=rand(2..N):
```

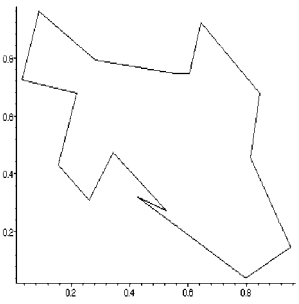
解答例

これだけのヒントで後は自由にやってもらいました。2001年度は時間中にある程度解けた学生が数名、最終的に10数名が解答を提出しました。解答例は以下のとおりです。

```
#2都市間の距離の計算
Distance:=Matrix(1..N,1..N,shape=symmetric):
for i from 1 to N do
  for j from i+1 to N do
    Distance[i,j]:=norm(Point[i]-Point[j]);
od;od;
#あるルートでの距離の和
E_sum:=tmp_a->add(Distance[tmp_a[i],tmp_a[i+1]],i=1..N);
tmp:=[];
T:=0.05;
max_iter:=10000;
E_a:=E_sum(a);#E(a)を求める
for iter from 1 to max_iter do
  sel_i:=Sel_city();          #2都市を選ぶ
  sel_j:=Sel_city();
  if (sel_i=sel_j) then next fi;
  del_a:=a;          #配置a+δaの生成
  del_a[sel_i]:=a[sel_j];
  del_a[sel_j]:=a[sel_i];
  E_da:=E_sum(del_a);
  dE:=E_da-E_a; #ΔEを求める
  if (exp(-dE/T)>evalf(rand()/10^12)) then
    a:=del_a;      #exp(-ΔE/T)の確率で受け入れる
    E_a:=E_da;
    tmp:=[op(tmp),E_a];
  end if;
od:
#ルートの変化に伴う距離の変化
listplot(tmp);
```



```
#最終的なルート
Route:=[seq(Point[a[i]],i=1..N),Point[1]]:
PLOT(CURVES(Route));
```



注：これは最短経路ではない。

3. 反省と分析

3.1 学生の評価

このような演習内容について受講生がどのようにとらえているかのアンケート結果をTable 2に示します。このアンケートは3回生で演習を受けたであろう大学院生に協力してもらいました。彼らは大学院で私が開講している計算材料学で強制的にMapleを使わされています。

結果は演習の当初目標とした“材料科学の解き方を知らす”にはある程度到達しているようです。しかし、Mapleの使用法の習熟度は決して高くありません。代表的な肯定意見は

数式を展開したい時や、直感で理解するためにグラフにしたい時など、ちょっとした事につかっていますが非常に便利です。

ですが、彼は3回生で受講していませんし、他のヘビーユーザーも私の指導学生達ですので、演習の授業で使えるようになったわけではありません。一方、否定的な意見は

一通りの使い方をわかっていれば Mapleは必要かつ頻繁に使うソフトになりますが、中途半端にしかわかっていないととっつきの悪いあまり使わないソフトになります。授業ではそのとっつきの悪さをいかに乗り越えさせるかが重要だと思います。その観点から見ると、今の授業はすぐに問題が難しくなり、Mapleに費やす時間よりも問題とその解答を理解するのに多くの時間が割かれ、実際にMapleをいじっている時間はわずかなように思われます。もっと基本的な問題を多く解かせ、材料科学的な問題は最後に少し示す程度でもいいのではないでしょうか。

に集約されています。この学生さんは優秀なほうで、意図せずに演習の設定目標にはまってくれています。

普通のレベルの学生さんでは、非常に早く課題を仕上げる者が増えましたが、単にタイプミスが減っただけなようです。テキスト[2]が完成して解答付きになったのがかえってあだとなり、テキスト通りの解がでたら満足してしまい、パラメータを変えたり、視覚化を試みたり別解を考えて遊ぶという学生はほとんどいません。せっかく便利な数式処理ソフトを試用するのですから、出来るだけ多くの学生に日常の学習に使ってほしいと願います。そこで数式処理ソフトを教えるという観点から、先程の学生さんが指摘している“とっつきの悪さ”の出所を分析します。

Table 2　アンケート結果

3回生のMapleの演習は役に立ったか？	
受講していない	2
覚えていない	8
全然役にたたなかった	1
材料科学の問題の解き方がわかった.	8
演習以外の問題をMapleで解いた	1
現在のMapleの使用状況	
計算材料学のレポート以外では使わない	9
ときどき触ってみるが，あまり使えない	5
使う必要を感じない	1
よく使っている	5
Maple(数式処理ソフト)は必要？	y.20 n.0
Mapleをもっと徹底的に教えるべき	y.12 n.8
Mapleの参考書がもっと必要?	y.17 n.3

3.2 Mapleのとっつきの悪さ

まず、Mapleのとっつきの悪さをより専門的なプログラミング言語、およびより一般的な表計算ソフト(Excel)と比較します。

Fortranと比較すると、数式処理ソフトでは最初に覚えるべきコマンドが多いという明白な“とっつきの悪さ”があります。しかし、そのおかげで具体的な例で示したように非常に短いスクリプトで目的とするグラフを得ることが可能です。

例えば先程の巡回セールスマン問題をUnix上のFortranで解かせようとすると、後処理に必要な知識として

unix上でのredirectionの考え方
mule(emacs)によるデータ加工
作図のためのgnuplotの使用法
出力のためのgv(ghost view)の使用法

などの説明が不可欠で、これだけの量を説明するとなると一コマが飛んでしまいます。実用的なプログラムを開発する際に必要となるライブラリー(NAGやNumerical recipe)の知識まで含めて比較すると、数式処理ソフトのコマンドはそれほどのオーバーヘッドになっていないと思います。

一方、Excelは初心者にとって確かに覚えやすいapplicationです。しかし、単純なグラフを書かすときにとるあの手間と数式処理ソフトのplotとを比べてどちらが使いやすいでしょうか? また、数式処理ソフトのユーザーインターフェースはとっつきの悪いキャラクターベースですが、解(solve)、展開(expand)あるいは分子/分母(numer/denom)等の単語は英語圏では数学用語として常識です。普通のレベルの学生でも、それほど苦もなく覚えたり推測したりできます。これらのとっつきの悪さは英語に起因しています。

最も根本的な違いは、ExcelやFortranでは数式処理が不可能だという点です。大ざっぱに言ってしまえば、Excelは四則演算、Fortranはプログラミングのための Application/言語ですが、Mapleは数学、数値計算、プログラミング、視覚化という非常に広い領域をターゲットにしています。前提とする知識量や、覚えておくべき最低限のコマンドの数も全然違ってきます。

とはいえ、初心者に対するしきい値を下げるために、数式処理ソフトの環境をより“とっつきやすく”する努力がまだまだ必要でしょう。

3.3 悩んでいるのは理論なのか操作法なのか

もう一つの“とっつきの悪さ”は実はMapleの便利さに由来しているようです。

数式処理を教えるのは、Fortranなどの言語でよく取り上げられるNewton-Raphson法や掃き出し法を使って数値計算やプログラミングを教えるのとは質的な違いを感じます。その程度の問題はコマンド一発で終ってしまうからです。

このような便利さがあだとなって、Mapleのコマンド入力にとられる時間と、背景知識の理解にとられる時間に大きな差が出ます。こういう状況に陥ると、Mapleの操作では全くつまずいていないのに、数式処理ソフトを習得したという気にはなかなかなれません。

3.4 もっと基礎的な演習を!?

では数式処理ソフトの習熟には、さきほどの優秀な学生さんが提案されているように、基礎的な問題をもっと多く解かせるのが上策なのでしょうか？　これに対する答えはNoです。

その理由は、Mapleなどの数式処理ソフトは基礎的な問題をいくらやってもレベルはあがらないという実体験にあります。入門書の厚さに先ず圧倒され、なんとか意を決して基礎的な問題を順番にやっても次々と出てくるコマンドは覚えられず、いざ問題を解こうとしても手をつけることすら出来ず、何も残らない時間を過ごした我が身を哀れみます。数式処理を自由に使いこなすというのは、高校の数学の問題集を解くのとは違ったアプローチが必要です。紙と鉛筆ではお手上げな、でも解かなければいけない問題を前に、個々の問題に分解してテキストのあちこちを調べながら解いていくのが一番の早道です。

実際、私の卒研生や大学院生は同じテキスト[2]で勉強して一週間程度でMapleを縦横に使いこなしています。これには脅し(でけへんだら面倒みんぞ)がだいぶ効いているようです。さらに研究の必要性から、継続してMapleを使う環境にあることが技能を高めています。

3回生の演習では中身への興味や、問題が解けないことのいらだち、卒研などの切迫感がありませんので、脅しの代わりになる“おいしいえさ”をうまく用意する必要がありそうです。

4. さいごに

Mapleなどの数式処理ソフトは、数学やプログラミングを専門としない研究者が日常の研究を進めるうえでは必須の道具であるとわたしは実感しています。また、理論とその応用を効率的に教え、それらを縦横に操る研究者を育てるうえでは最適な道具と判断しています。もちろんこれは数学や数値計算を専門とする分野でも一部あてはまると思います。あまり認めたくないのですが、今の学生さんには数式処理ソフトのどこがMagicなのかちゃんと伝わってないかもしれません。『もしも私がこれだけ便利なソフトを学生の時に知っていたら、もっといろいろな分野や問題で遊べたのに』という思いがあり、今の学生さんには何とか使ってほしいと強く願います。ただ数式処理ソフトの使用法は誰かに教えてもらったのでなく、30代になってからテキストで独習しました。したがって、若い頭脳への教授法や、トピックスの選び方に間違いや偏りがあるかもしれません。この記事に対してご批判、ご意見をいただければ幸いです。

参考文献

[1] 梶原健司、「エンドユーザの賢い数式処理ソフト選び」、数理科学、No.425, 1998/11, pp.39-44.

[2] 西谷滋人、「利用の手引きーMaple編ー」、2001、京都大学大型計算機センター。
URL http://www.mtl.kyoto-u.ac.jp/users/bob/Maple/Maple.html

[3] 例えば、ニーナ・ホール編「カオスの素顔」Blue Backs P880, 講談社、1994にはいくつものCGとともにMandelbrot自身が語るフラクタルの解説が10章にあり、キャロライン・シリーズによる11章はフラクタル幾何学の直観的な理解を助けてくれる。

[4] 奥村晴彦、「C言語によるアルゴリズム事典」、技術評論社 1991

著者紹介
西谷　滋人
所属：京都大学工学研究科材料工学専攻
専門：第一原理計算，分子動力学による金属，セラミックス材料の格子欠陥，組織形成シミュレーション
e-mail:nishitani@mtl.kyoto-u.ac.jp,
http://www.mtl.kyoto-u.ac.jp/users/bob