解説

MATLABクローンによる大域的最適化(1)
—Octaveに何ができるか—

久野　誉人

1. はじめに

春秋の研究発表会などで最適化関連のセッションに出席すれば，そこで提案される様々な新しいアルゴリズムの性能評価には最悪計算量の解析だけでなく，今やコンピュータを用いた数値実験が常識のように行われている．かつては，この数値実験に用いるプログラムの作成にアルゴリズムそのものの設計に要した時間の5〜10倍もかかっていたので，「いったい自分の研究はORなのか，それともプログラミングなのか」と卒研や修論のときに嘆いた読者も多いに違いない．ところが，最近はFortranやCなどの高級言語を用いた手作りのプログラムは滅多に見かけなくなり，もっぱらソルバーと呼ばれるCPLEX[13]やXpress-MP[10]などが流用され，実験にかかる手間は驚くほど軽くなっている．また，ソルバーが使えないような凝ったアルゴリズムを作っても，MATLAB[9]という便利な道具もあるので，まるでレゴ・ブロックでも組み立てるみたいにアルゴリズムをプログラム化できてしまう．こういう夢のような環境を当り前のように享受している学生たちには嫉妬を禁じえないが，「待てよ，CPLEXにしろMATLABにしろ商用ソフトで，しかも相当に高額だけれど，筆者のような甲斐性なしの指導教官に就いてしまった学生達は大丈夫なのかしらん？」と心配にもなるのである．学生と言えどもパソコンくらいは自宅に持てる時代だが，その中に夢の計算環境を実現できるほどリッチな学生がいるとも思えず，結局自宅のパソコンはインターネット検索とワープロくらいにしか使っていない，というあたりが落ちだろう．

今月号から三回にわたって，自宅のパソコンに夢の計算環境を無料で実現し，しかも難しい非凸計画問題を大域的に解決するためのプログラムを簡単に作成する方法について解説していく．用意してほしいのは，インターネットに接続されたパソコン1台と，若干のオタクな情熱だけである．その二つさえあれば，フリーソフトウェアを使うことで，少なくとも自分が設計したアルゴリズムの善し悪しの見極めに指導教官の懐具合まで気にしないで済むのである．初回は，そのフリーソフトウェアGNU Octave[15]の導入と使いかたを解説する．第2節で，Octaveの生い立ちについて紹介し，自宅のパソコンに導入する方法を簡単に説明したのち，第3節ではOctaveに何ができて，何ができないかを検証する．第4節では，次回の予告編もかね，Octaveを使って線形計画問題を解く改訂単体法のプログラム作成を読者に宿題として出すつもりだ．宿題の解答は，次回，詳細にわたって解説する予定だが，どうして内点法ではなく単体法なのかというと，最終回に取り上げる非凸計画問題の大域的最適解を求める分枝限定法の中で，線形計画問題に対する感度分析を繰り返し行う必要があり，それを宿題のプログラムで実行しようという魂胆である．Octaveの使い勝手のよさを十分に体感したところで，最終回にはOctave自体の高速化の方法などについても紹介するつもりである．

くの　たかひと
筑波大学 大学院システム情報工学研究科
〒305-8573 つくば市天王台1-1-1

2. Octaveは自由だ

GNU Octaveは，Wisconsin大学化学工学科のJohn W. Eatonらが中心となって開発，管理している数値計算に特化した言語で，使用感はFortranやCよりもむしろunixのshellに近い．線形・非線形システムを数値的に処理するための便利なコマンドライン・インターフェースが用意してあり，そこから簡単な文法を使って様々な数値実験を実行することができる．その文法がMATLABと非常に高い互換性があるため，Octaveは一般にMATLABのクローンと見なされている．MATLABクローンと呼ばれるフリ

ーソフトウェアには他に，フランス国立情報・自動制御研究所のScilab[8]が有名で，MATLABに引けをとらない綺麗な描画ツールとあいまって人気も高い．Octaveにも描画ツールは用意されているのだが，gnuplot[12]という既成の地味なソフトをそのまま流用しているため，MATLABやScilabに比べてかなり見劣ることは否めない．しかしOctaveは，線形計算ライブラリーのBLASとLAPACK[14]をチューン・アップすることで容易に高速化でき，そして何よりフリーソフトウェア財団のGNU一般公衆利用許諾契約書（GPL）[11]に従って自由に再頒布や修正の行える点が大きな魅力である．そう，Octaveに対して用いるフリーソフトウェアという言葉には，無料ソフトの意味のほかに，自由なソフトという意味が込められているのである．

ところで，開発責任者であるEatonの所属が，情報工学やコンピュータ・サイエンスではなく，化学工学科というところに違和感はないだろうか．実は，化学工学の学生たちもシミュレーションなどで我々と同じようにFortranやCによるプログラミングには泣かされていて，彼らには常に「バグ取りばかりで肝心の化学工学を勉強する時間がない」という不満があるらしい．学生たちの不満を解消するため，反応装置工学に関する教科書[3]用の「あんちょこ」として80年代後半に企画されたソフトウェアがOctaveの前身である．この話を聞けば，Eatonの専門が化学工学であっても，その趣旨にはORを専門とする我々も共感が持てるし，Octaveも何だか安心して使えるでしょ？

さてOctaveの入手方法だが，そのホームページ[15]から安定版（最新はoctve-2.0.17），テスト版（同octave-2.1.71），開発版（同octave-2.9.3）のソースをいずれも圧縮したtarファイルoctave-*.*.*.tar.gzやoctave-*.*.*.tar.bz2の形でダウンロードできる．これを解凍するとoctave-*.*.*という名前のディレクトリー（フォルダー）ができるので，その中にあるインストラクションREADME.*，INSTALL.*に従い，GCC（GNU Compiler Collection）[6]でコンパイル/インストールすればLinuxやWindowsなど大抵のOS上でOctaveを動かすことができる．この辺りの情報は既にインターネットに溢れているので，信頼できるホームページを参考に作業を進めるとよいだろう．なお，DebianやFedora Core（Red Hatの無償版），Mandriva（旧Mandrake）などのメジャーなLinuxにはOctaveのバイナリ・パッケージがオプションで付属しているので，Linuxユーザならば即座に使いはじめることができるし，流行のlive-CD Linuxの一つKNOPPIX/Math[7]の中にはOctaveが導入済みであるので，Windowsユーザであってもその環境に手を加えることなく試してみることは可能だ．

ちなみに筆者の環境はIBMのThinkPad X40（Pentium M, 1.10 GHz）にインストールしたFedora Core Linuxであり，そのバイナリ・パッケージであるoctave-2.1.57.i386.rpm[5]の使用を想定して話を進めて行くことにする．これはテスト版だが，安定版のoctave-2.0.*は一番新しいものでもリリースが2002年と古く，色々な面で使いづらい．しかも，コンパイルに一世代前のGCC 2.95のC++が必要なので，これを標準的なGCC 3.*と共存させようとするとかなり厄介な作業が必要になる．メジャーなLinuxの多くに付属するOctaveのパッケージもテスト版のいずれかであり，octave-2.1.*の導入がお薦めである．

3. Octave事始め

Octaveのインストールは無事に終了しただろうか．では，さっそくOctaveを走らせてみよう．

OSのコマンドライン（プロンプトを$で示す）から「octave」と入力すると，

```
$ octave
GNU Octave, version 2.1.57 (i686-pc-linux-gnu).
Copyright (C) 2004 John W. Eaton.
This is free software; see the source code for copying conditons.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTIBILITY or
FITNESS FOR A PARTICULAR PURPOSE. For details, type 'warranty'.
.....                (省略)                .....
octave:1>
```

のようにライセンスに関するメッセージののちにOctaveのプロンプト「octave:1>」が現れ，Octaveの世界への扉が開かれる．しかし，この「ABOLUTELY NO WARRANTY」に怯んで使用をひかえてしまうしまう人も，特に企業に勤める人の中にはいるかもしれない．GNUのGPLは，そのソフトウェアの複製・頒布の自由を代価なしに保証する一方，正しく機能しなかった場合の保証は一切しない．必要な保守点検や補修，修正に要するコストはすべて

ユーザ自身が引き受けることになる．とは言え，GNUのソフトウェアは世界中のユーザから寄せられるバグ情報やそのパッチファイルによって頻繁に更新されているので，並の商用ソフトよりもはるかに安定しており，「Octaveで行った計算が間違っていた」なんてことはありえないので安心してよい．

3.1 簡単な行列演算

OctaveやMATLAB，そのクローンたちの大きな特徴は，サイズや成分のデータ型などを一々宣言しなくても行列をそのまま入力して操作できる点にある．例えば，Octaveのコマンドラインから

```
octave:1> A=[1 2;3 4]
```

と打ち込むだけで，

```
A=
   1  2
   3  4
```

のような確認の返事とともに行列

$$A=\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}$$

がOctave内に定義される．この返事は鬱陶しいこともあるが，入力の最後にセミコロン「；」を付けることで出力は止められる．行列Aが正しく定義されているか，その第(2,1)成分を出力してみよう：

```
octave:2> A (2,1)
ans=3
```

ベクトルは特殊な行列なので，Aと同様に

```
octave:3> b=[5;6], c=[7  8]
b=
   5
   6
c=
   7  8
```

とやれば，列ベクトルb＝$(5,6)^{\top}$も行ベクトルc＝(7,8)も定義できる．こうして定義したA，b，cにプライム「'」を付ければ転置が行われるし，和や積はforループなどを使わずに計算することができる：

```
octave:4> b'+c
ans=
   12  14
octave:5> x=A * b, y=c * A
x=
   17
   39
y=
   31  46
```

それればかりか，Ax＝bや$y^{\top}A=c$解x，yも，Aが正則なら，まるでスカラーどうしの演算感覚で求められるのである：

```
octave:6> x=A\b, y=c/A
x=
   -4.0000
    4.5000
y=
   -2  3
```

整数ばかりが続いたところで，いきなり−4.0000や4.5000のように小数点以下にゼロがいっぱい現れて「おや？」と感じるかもしれないが，データ型のないOctaveでは整数も有理数もすべて倍精度有理数として扱われる．

3.2 ループは使えるか

「和や積はforループなどを使わずに」と書いたが，Octaveにループの類がないわけではない．しかし，可能なかぎり使わない方がよいのである．そのことを示すために，もっと大きな行列を定義しておこう：

```
octave:7> B=rand(200,500); d=rand(500,1);
```

これで組込み関数rand()によって区間[0.0, 1.0]の一様乱数を成分とする200×500の行列Bと500次の列ベクトルdが定義されるが，入力の区切りを「,」から「;」に変更し，入力の最後にも「;」を付けることを忘れないように！　さもないと，しばらくの間Octaveからの返事で端末を流れる乱数の滝を眺める羽目になる．行列Bとベクトルdの積Bdは，forループを使えば次のような方法で計算できる：

```
octave:8> t=time(); x=zeros(200,1);\
>for j=[1:200]
>for k=[1:500]
>x(j)+=B(j, k) * d(k);
>endfor
>endfor; time()-t
octave:9> ans=3.4821
```

最初の行のtime()は現在の時間を秒単位で告げる組込み関数で，zeros()はすべての成分がゼロの行列を生成する組込み関数．したがって，ここでBdの計算結果を収める200次列ベクトルxを初期化している．この行の最後に付けた「\」は，Octaveへの命令が改行後も続くことを示している．次の行が，Bの行に関するforループの始まりで，Octave流に解釈するなら，「1から200までの数を成分とする200次行ベクトル[1:200]の各成分を順にjに代入し，forから対応するendforまでの命令を実行しなさい」という意味である．3行目から5行目までがBの列に関する

forループになり，Bの第j行とdとの内積がxの第j行に格納される．4行目の「x(j)+=B(j,k)*d(k)」はC言語のように「x(j)=x(j)+B(j,k)*d(k)」と同じ意味だが，安定版（octave-2.0.*）やMATLABでは使えないので注意が必要である．最後の行で，このプログラムの開始時間tと現在時間time()との差が取られており，セミコロン「;」が付いていないのでその結果，つまりプログラムの実行時間3.4821秒が出力される．

すでに見たとおり，Octaveではこんな複雑なことをしなくともBdは計算できるが，time()を使って計算時間も計測すると，

```
octave:10> t=time(); y=B * d; time()-t
ans=0.00064802
```

のようにわずか0.0007秒（!!）足らずしか掛からないことがわかる．「同じBdの計算に3.4821秒と0.0007秒とでは差がありすぎる．本当に正しく計算できているのか？」と疑い深い人のために，ループを使って計算した結果x=Bdと，使わずに計算した結果y=Bdを比較してみよう：

```
octave:11> find (x!=y)
ans=[ ](0×1)
```

ここで，「x!=y」はベクトルxとyを成分ごとに比較し，異なっていれば1，同じならば0を成分とする列ベクトルを生成する．組込み関数find()は，さらにその成分からゼロでないものをすべて見つけ出し，行番号を列ベクトルにして出力する．したがって，もしxとyの第j行が異なっていれば，「find(x!=y)」はjを含むベクトルを返すはずであるが，結果は行数ゼロの列ベクトル[ ](0×1)，つまり二つの計算結果xとyは完全に一致したということである．結果が同じで計算時間が5,000倍も違うのだから，どちらの計算方法を使うべきか，いや，使ってはいけないのか明らかであろう．

厳密に言えば，time()で計測した時間にはOSがOctaveとは別にバックグラウンドで行っている作業の時間も含まれるので，実際の計算時間に5,000倍もの差はないだろう．しかし，そうだとしても小手先の工夫で解消できるような差であるはずがない．このスカラー演算をループで繰り返したときの効率の悪さは，Octaveに限ったものではなく，MATLABやそのクローンたちに共通する属性である．

3.3 関数を作る

使ってはいけない計算方法ではあるが，「quit」と入力すると，

```
octave:12> quit
$
```

のようにOctaveの終了と同時に消えてなくなり，いささかもったいなくもある．しかし，テキストファフィルに関数として保存しておけば，そのファイルが保存されたディレクトリでOctaveを起動したときにはいつでも実行することができる：

```
function x=product(B, d)
    [m, n]=size(B);
    x=zeros(m, 1);
    for j=[1 : m]
      for k=[1 : n]
        x(j)+=B(j, k) * d(k);
      endfor
    endfor
endfunction
```

最初の行のfunctionは，BとdをA力としてxを出力する関数product()の定義の始まりを示しており，これに対応する最後の行のendfunctionが定義の終わりである．2行目のsize()は行列のサイズを返す組込み関数で，入力された行列Bの行数と列数がそれぞれmとnに格納される．残りの行は，節3.2の例とほぼ同じであるので説明は不要だろう．関数名が組込みのものとかち合わないように注意して，それに拡張子「.m」を付けたものをファイル名にする．したがって，いまの場合，関数product()はファイルproduct.mに保存する．もう一度Octaveを立ち上げ，ランダムに作った行列とベクトルに対して関数product()を実行してみよう：

```
$ octave
GNU Octave, version 2.1.57 (i 686-pc-linux-gnu).
.....                (省略)                .....
octave:1> A=rand(400,300); b=rand(300,1);
octave:2> t=time(); x=product(A, b); time()-t
ans=4.0034
octave:3> t=time(); y=A * b; time()-t
ans=0.00079012
ctave:4> find(x!=y)
ans=[ ](0×1)
```

組込み関数rand()を使って改めて定義した行列Aとbが関数product()に渡されると，product()はその定義の中のBとdにそれぞれを代入して積Bdを計算し，その値をx=Abとして返す．しかし，関数として保存したところでOctaveが実行するのは節3.2と同じ演算なので，「y=A * b」による計算時間にか

なうはずはない．関数product()は，あくまでOctaveの使い方の練習であって，やはり使ってはいけない関数なのである．

ところで，forループで用いるカウンタにはFortranやCでは「i」を用いることが多いが，

```
octave:5> i, x=i * i, e, pi
i=0+1i
x=-1
e=2.7183
pi=3.1416
```

とやってみればわかるように「e, pi」などと並んで予約語になっているで，カウンタとしてはこれも使わない方がよい．

3.4 その他の操作と演算

ループがOctaveの禁じ手となるとFortranやCのような発想ではプログラミングが非常に難しくなるが，Octaveにはループを使わなくても済むように便利な操作や演算がいくつも用意されている．例えば，先ほど定義した400×300の行列Aや300次ベクトル列bから，次のように部分行列，部分ベクトルを抽出することができる：

```
octave:6> B=A(5:7, 200:2:208)
B=
    0.760686   0.719640   0.714155   0.926419
    0.639348   0.735530   0.093768   0.402434
    0.857055   0.122170   0.695964   0.728949
    0.284629   0.868804   0.836996
octave:7> c=A(5, 208:-2:200), d=b(1:5:25)'
c=
    0.63935   0.92642   0.71416   0.71964   0.76069
d=
    0.62444   0.45488   0.67625   0.73654   0.88933
```

行列BはAの第5,6,7行の第200列から1列飛ばしに第208列までの5列から構成される部分行列で，cはAの第5行の第208列から−1列飛ばしに第200列までの5成分で構成される行ベクトル，つまりBの第1行を逆順に並べたものに等しい．ベクトルdは，列ベクトルbの第1成分から四つ飛ばしに第25成分まで，つまり第1,6,11,16,21成分からなるベクトルの転置である．サイズが等しいc，dに対して

```
octave:8> x=c.* d, y=c./d, z=c.^2
```

のように「.」を付けて演算を行うと，成分ごとの演算結果が同じサイズのベクトルとして出力される：

```
x=
    0.39924   0.42141   0.48295   0.53004   0.67650
y=
    1.02387   2.03661   1.05605   0.97706   0.85535
z=
    0.40877   0.85825   0.51002   0.51788   0.57864
```

また，組込み関数sum(), max(), min()はそれぞれ入力となる行列の各列成分の和，最大値，最小値を行ベクトルで返す：

```
octave:9> x=sum(B), y=max(B), z=min(B)
x=
    2.1922   1.5424   1.4012   2.6523   1.5985
y=
    0.76069   0.72895   0.71416   0.92642   0.83700
z=
    0.695964   0.093768   0.284629   0.857055   0.122170
```

これらの出力を再度，sum(), max(), min()の入力とすれば，

```
octave:10> x=sum(x), y=max(y), z=min(z)
x=9.3865
y=0.92642
z=0.093768
```

のように行列Bの成分すべての和，最大成分，最小成分がえられる．したがって，ベクトルcとdの成分ごとの積c.* dをsum()に入力すればcとdとの内積が返される：

```
octave:11> x=sum(c.* d), y=c * d'
x=2.5101
y=2.5101
```

成分ごとの演算子「.*」と比較演算を用いれば，最大値や最小値だけでなく，例えば行列Bから値が0.5以下の成分を抽出することもできる．つまり，行列Bとスカラー0.5を比較することで

```
octave:12> C=(B<=0.5)
C=
    0  0  0  0  0
    0  1  1  0  1
    0  0  1  0  0
```

のようにBの0.5以下の成分には1，他の成分に0が対応する行列Cが生成される．このCとBとの成分ごとの積を取れば

```
octave:13> B.* C
ans=
    0.00000   0.00000   0.00000   0.00000   0.00000
    0.00000   0.09377   0.40243   0.00000   0.12217
    0.00000   0.00000   0.28463   0.00000   0.00000
```

のようにBの0.5より大きな値の成分はゼロに置き換えられる．また，0-1行列Cを直接Bに引数とし

て渡せば，

```
octave:14> B(C)
ans=
    0.093768
    0.402434
    0.284629
    0.122170
```

のようにBから0.5以下の成分だけを列ベクトルにして抽出できる．

FortranやCの発想でB(C)を実行しようとすると，ループやif文を使って，例えば次のような関数を作ることになる：

```
function x=lehalf(B)
    [m, n]=size(B);
    B=reshape(B, m * n, 1);
    j=1; x=[ ];
    while j<=m * n
      if B(j)<=0.5
        x=[x; B(j)];
      endif
      j++;
    endwhile
endfunction
```

実は，この関数lehalf()の中でも，3行目のreshape()で行列Bをmn行1列，つまり列ベクトルに書き換えたり，7行目でベクトルxの成分を追加するなどOctave流の操作を行っている．また，「j++」は「j=j+1」と同じ意味だが，関数product()で用いた「x(j)+=B(j, k) * d(k)」と同様にC言語風の文法で安定版のOctaveやMATLABには使えない．いずれにしても，lehalf()で400×300の行列Aなどから0.5以下の成分を抽出するようなことはやめた方が賢明だ：

```
octave:15> t=time(); x=lehalf(A); time()-t
ans=63.685
octave:16> t=time(); y=A(A<=0.5); time()-t
ans=0.0093331
octave:17> find(x!=y)
and=[](0×1)
```

Octaveには他にもまだまだ面白い使い方があり，パソコン上で色々と試してみると四五日は遊んでいられる．これらを上手に組み合わせることで，最適化アルゴリズムに必要なほとんどの操作はループなんて使わなくてもできることがわかるはずだ．残念ながら紙面の都合もあり，そのすべてをここで紹介することはできないが，残りの使い方に関してはOctaveのマニュアル[15]を参照しながら読者自身で試して欲しい．

4. 読者への宿題

最後に，次回までの宿題を読者に出しておこう：

宿題1. 線形計画問題

$$\left|\begin{array}{ll} \text{最大化} & c^{\top}x \\ \text{条　件} & Ax\leq b,\ x\geq O \end{array}\right. \tag{1}$$

を解くための改訂単体法をOctaveかMATLABを使ってプログラム化しなさい．ただし，$A\in\mathbb{R}^{m\times n}$, $c\in\mathbb{R}^{n}$とし，$b\in\mathbb{R}^{m}$の成分はすべて非負とする． ■

制約条件の右辺定数bが非負ベクトルなので，$x=O$は問題(1)の実行可能解である．したがって，スラック変数を基底変数x_B，元の変数を非基底変数x_Nとし，単位行列を$I\in\mathbb{R}^{m\times m}$で表すことにして

$$B=I,\ N=A,\ c_B=O^{\top},\ c_N=c^{\top}$$

と置けば，ただちに実行可能な辞書

$$\left|\begin{array}{l} x_B=B^{-1}b-B^{-1}Nx_N \\ z=c_BB^{-1}b+(c_N-c_BB^{-1}N)x_N \end{array}\right. \tag{2}$$

が得られ，改訂単体法のフェイズIIだけをプログラムにすればよいことがわかる[1, 2, 4]．しかし，どんな最適化アルゴリズムも反復によって解を更新していくので，たとえOctaveと言えどもループなしにこれをプログラム化することは不可能だろう．

参考文献

[1] Chvátal, V., *Linear Programming*, Freeman (1983).
[2] 今野　浩,「線形計画法」, 日科技連 (1987).
[3] Rawlings, J. B., and J. G. Ekerdt, *Chemical Reactor Analysis and Design Fundamentals*, Nob Hill Publishing (2002).
[4] 田村, 村松,「最適化法」, 共立出版 (2002).
[5] http://download.fedora.redhat.com/
[6] http://gcc.gnu.org/
[7] http://geom.math.metro-u.ac.jp/wiki/
[8] http://scilabsoft.inria.fr/
[9] http://www.cybemet.co.jp/matlab/matlab
[10] http://www.dashopt.com/
[11] http://www.gnu.org/
[12] http://www.gnuplot.info/
[13] http://www.ilog.com/
[14] http://www.netlib.org/
[15] http://www.octave.org/