本稿は Linux Japan 誌 2001 年 5 月号に掲載された記事に補筆修正したものです.

任意精度数値計算ツール

Linux 上の数式処理ツールに関するお話の続きです. 前回は, 任意精度数値計算ライブラリの話で終りました. 今回は, 数学の王国フランス産まれの PARI+GP[1]W[3] という軽くて使い勝手の良い任意精度数値計算ツールを紹介します.

インストール

ちょっとだけてこずるところがあるので, インストールに関する注意を述べておきます. 通常の手順にのっとって, ソース pari-2.1.0.tar.gz を展開し, "`./Configure -ask`" に対して, 何も考えず (もちろん考えて答えてもいいです) Enter を押していけば Makefile が出来てしまいます. ただし, この時点で make すると Plamo2.1 ではドキュメント作成の際に `eufm10.tfm` ... などが無いと怒られました. 実行ファイルはできているのでこれでも問題ありませんが, ちゃんと最後まで終えいという方は, これらの飾り文字フォントの `*.tfm` と `*.mf` を `CTAN/fonts/amsfonts/`[2]W[3] から入手して, それぞれ `$TEXMF/fonts/tfm/ams/` `$TEXMF/fonts/source/ams/` というディレクトリにでも入れておいてください (念のため, おまじない `mktexlsr` をお忘れなく). この準備の後, `make gp`, `make install` で最後までインストールが進むと思います.

GP の特徴

有理数演算

GP は 任意精度数値計算ライブラリ PARI を応用した数式処理ツールです. `gp` を起動すると, バージョン情報などが表示された後, プロンプトが現れますからコマンドを入力していきます.「さすが数式処理ツール普通の電卓と違うわい」と感じる部分から紹介しましょう

```
(09:51) gp > x = 1/3 + 1/2
%1 = 5/6
(09:53) gp > y = 1/11 + 1/5
%2 = 16/55
(09:53) gp > x+y
%3 = 371/330
(09:55) gp > x*y
%4 = 8/33
(09:55) gp > x^2
%5 = 25/36
(09:55) gp > \q
Good bye!
```

分数の計算を分数のまま実行してますね. 分数計算のできない大学生の自習用にどうぞ. 冗談はさておき, 単なる数の計算ではなく, 数の概念を確かめるものですから, ライブラリの段階で既に有理数というクラスを定義しているのです (前回紹介した CLN も同様です). "`\q`" または "`quit`" で終了します. GP はオンラインヘルプが充実しています. "`?`" と入力すると, tutorial や refcard の呼び出し方法まで含めた項目が表示されます. 上記の有理数の演算は "`??  tutorial`" で呼び出された tutorial.dvi (46 頁と手頃な長さ) の最初に書いてある事柄です. この tutorial は大変面白いのですが, 段々と数学専門の難しい話題もでてくるので, 中学・高校や大学の初年度に習う事柄の扱いについてだけ紹介します.

因数分解

まずは, 懐かしい**因数分解**です.

```
(22:46) gp > default(prompt,"gp > ")
   prompt = "gp > "
gp > \o 0
   output = 0 (raw)
gp > factor(5940)
%1 = [2, 2; 3, 3; 5, 1; 11, 1]
gp > factor(x^6-x^5-x+1)
%2 = [x - 1, 2; x^4 + x^3 + x^2 + x + 1, 1]
gp >
```

プロンプトがうるさいので `default(`*def,value*`)` 命令で変更しました. また紙面節約のために, 出力形式も簡単なものにしました. これらのコマンドは `?default ?\o` で確認できます. 以降の表示は, 初期化ファイル `~/.gprc` に

```
prompt = "\e[1mgp\e[m > "
output = 0
colors = "no,7,no,no,no,no,no"
```

と設定した場合のものです (colors の 2 番目はヒストリ番号の色です. もちろん実際には表示されますが, 画面上めだたないようにしています). 本筋に戻って, 実例の通り, $660 = 2^2 \times 3^3 \times 5^1 \times 11^1$ であり, $x^6 - x^5 - x + 1 =$

$(x-1)^2(x^4+x^3+x^2+x+1)$ であることがすぐ判ってしまいます．整数と多項式について同じ関数名であるところが当り前なのでしょうが，感心しました．

複素数

次に複素数です．GP では純虚数 $\sqrt{-1}$ を“I”で表します．

```
gp > (1+2*I)/(3+4*I)
     11/25 + 2/25*I
gp > \p 16
   realprecision = 19 significant digits
   (16 digits displayed)
gp > cos(1+I)
     0.8337300251311490 - 0.9888977057628651*I
gp > cos(I*Pi)
     11.59195327552152 + 0.E-18*I
gp > cosh(Pi)
     11.59195327552152
```

紙面の都合で精度を 16 桁に落しました (既定は 28 桁です)．複素数上でも有理数演算が継承されています．また，$\cos(i\pi)=\cosh(\pi)$ がきちんと示されました．ここまでは当り前ですが，次はどうでしょう．

```
gp > \o 2
   output = 2 (prettyprint)
gp > polroots(x^3+1)

[(-1.000000000000000 + 0.E-19 I)]
[(0.5000000000000000 - 0.8660254037844386 I)]
[(0.5000000000000000 + 0.8660254037844386 I)]
```

紙面が狭いので出力形式を変えて表示させましたが，$x^3+1=0$ の 3 根を複素数根を含めて求めることができました．もちろん $x^3+(2+3i)x+4+i=0$ などの複素係数の多項式も OK です．

多項式

三角関数や指数関数 $f(x)$ は冪関数 x^k の無限級数和

$$f(x)=\sum_{k=0}^{\infty} c_n x^k$$

と表現されます．これにより，四則演算しか知らないコンピュータ（実は人間も）が超越関数の値を計算できるようになるわけで，極めて重要な考え方です．GP では関数全体この概念を通して把握できるようにいろいろなコマンドが用意されています．例えば，三角関数は

```
gp > sin(x)
     x - 1/6*x^3 + 1/120*x^5 - 1/5040*x^7 +
1/362880*x^9 - 1/39916800*x^11 + 1/6227020800*x^13
- 1/1307674368000*x^15 + O(x^16)
```

のように表されます．項数を無限に示すことはできないので，有限の項数で打ち切ります (default(seriesprecision,n) で設定します) が，人間が筆算で，あるいはその延長上にあるコンピュータが四則演算を用いて，実際に用いる計算式が表現されます．$\sin(x)$ の冪級数展開は理工系の学生ならば誰でも知っている筈ですが，$1/\sin(x)$ となると難しいでしょう．GP ではあっさり

```
gp > default(seriesprecision,8)
   seriesprecision = 8 significant terms
gp > sin(x)
     x - 1/6*x^3 + 1/120*x^5 - 1/5040*x^7 +
O(x^8)
> 1/sin(x)
     x^-1 + 1/6*x + 7/360*x^3 + 31/15120*x^5 +
O(x^6)
gp > sin(x)*1/sin(x)
     1 + O(x^7)
```

と答えてくれます．$1/\sin(x)$ と全く違うのですが，表記上間違えやすい $\sin^{-1}(x)=\arcsin(x)$（逆三角関数）は一般に標準的に用意されている数学関数ですが,「冪級数展開された関数の逆関数の冪級数展開を求める」という GP 特有のコマンドがあります．

```
gp > asin(x)
     x + 1/6*x^3 + 3/40*x^5 + 5/112*x^7 + O(x^8)
gp > serreverse(sin(x))
     x + 1/6*x^3 + 3/40*x^5 + 5/112*x^7 + O(x^8)
```

確かに，$\arcsin(x)$ と同じ冪級数展開となってます．本当に逆関数となっているか，一般の関数について例を示しましょう．

```
gp > f=x/(1-x^2)^(1/3)
     x + 1/3*x^3 + 2/9*x^5 + 14/81*x^7 + O(x^9)
gp > g=serreverse(f)
      x - 1/3*x^3 + 1/9*x^5 - 2/81*x^7 + O(x^9)
gp > subst(f,x,g)
      x + O(x^9)
```

subst(f,x,g) は関数 $f(x)$ の変数 x を式 g で置き換えるコマンドです．$g=f^{-1}$ ならば逆関数の定義より $f(g(x))=f(f^{-1}(x))=x$ となるのは当然ですね．

行列演算

GP は Octave や Scilab に比べて行列に関する演算が貧弱です．しかし，代数的な扱いが徹底している点が評価できます．例えば，行列 A の逆行列 A^{-1} は，Octave などでは inv(A) と表しますが，GP では

```
gp > A=[1,2;3,4]
     [1, 2; 3, 4]
gp > 1/A
     [-2, 1; 3/2, -1/2]
gp > A^-1
     [-2, 1; 3/2, -1/2]
gp > \o 2
   output = 2 (prettyprint)
gp > \p 16
   realprecision = 19 significant digits...
gp > 1./A

[-2.000000000000000 1.000000000000000]

[1.500000000000000 -0.5000000000000000]
```

と A^{-1} あるいは，何もそこまでと感心させられる $1/A$ で書き表せます．もちろん有理数演算は保たれていますし，小数点数表示もできます．従って，連立一次方程式 $Ax = b$ $(x, b$ は縦ベクトル$)$ の解は

```
gp > A=[1,2;3,4]
     [1, 2; 3, 4]
gp > b=[1,3]~
     [1, 3]~
gp > x=A^-1*b
     [1, 0]~
gp > 1/A*b
     [1, 0]~
gp > A*x
     [1, 3]~
```

のように式そのままの形で求められます．なお，関数 matsolve(A,b) も提供されています．

行列の固有値問題はどうでしょうか？ 関数 mateigen(A) がそれですが，有理数演算で頑張るため，次元の大きいものは解けなくなります．実対称行列に対する qfjacobi(A) はどうやら小数点数演算を行うらしく，大きな次元のものでも解けます．ただし，固有値 (と固有ベクトル) が大きさの順に整列されないので，ちょっと不便です．

```
gp > A=[1,0,0;0,2,0;0,0,3]
     [1, 0, 0; 0, 2, 0; 0, 0, 3]
gp > mateigen(A)
     [1, 0, 0; 0, 1, 0; 0, 0, 1]
gp > \o2
   output = 2 (prettyprint)
gp > \p 16
   realprecision = 19 significant digits ...
gp > qfjacobi(A)
     [
[1.000000000000000]
[2.000000000000000]
[3.000000000000000]
,
[1.000000000000000 0.E-19 0.E-19]

[0.E-19 1.000000000000000 0.E-19]

[0.E-19 0.E-19 1.000000000000000]
]
```

GP プログラミング

GP は対話的に使うことに主眼が置かれていますが，もちろんスクリプトを書いて実行させることもできます．対話的に実行するコマンドをそのまま記述すればよいのですが，少し面倒なことがあります．それは，計算結果が即時表示されてしまうことです．これでは大きな行列を定義したりすると，計算の度にそれが表示されてしまい時間を浪費します．この問題はコマンドの後ろにセミコロン ‘‘;’’ を付けることで回避できます (この手のツールでは常識らしい)．それでは量子力学の問題のなかから，無限深さの井戸 (幅=1) の中央に有限高さのポテンシャル (幅=1/8) がある場合の，固有波動関数を求める問題を解いてみましょう．数学的な背景は，一言でいうと，偏微分方程式を差分方程式に変換し，行列の固有値問題に帰着させて解くという至極常識的なもの (理工系の学生にとっては) です．リスト 1 を見てください．GP プログラムについて簡単に説明します．なお，GP ではスクリプトファイル名の拡張子は ‘‘.gp’’ とすべきなのですが (emacs 用のパッケージなどで限定されます)，gnuplot と重複してしまいますので，ここでは PARI からもじって .par としました．

リスト 1 eigen.par

```
 1: default(format,"f6.6");
 2: N=64; N1=N-1; VW=200.0;
 3: XM=1.0; W=1/16; h=XM/N;
 4: VD=2/h/h; VDT=-1/h/h;
 5: H=matrix(N1,N1,i,j,\
 6:   if(j==i-1|j==i+1, VDT, if(j==i, VD, 0));\
 7: );
 8: for (i=1,N1, if(abs(i*h-0.5)<=W, H[i,i]+=VW))
 9: X=qfjacobi(H);
10: V=X[2];
11: E=X[1];
12: ES=vecsort(E);
13: for(i=1,N1,\
14:   if(E[i]==ES[1], i1=i);\
15:   if(E[i]==ES[2], i2=i)\
16: )
17: for(j=1,N1,\
18:   print((j-0.)/N," ",V[j,i1]," ", V[j,i2])\
19: )
```

qfjacobi(H) で得られた固有値ベクトル E(=X[1]) は大きさの順に整列していませんので，vecsort(E) で整列させ，小さい値に対応する列 (i1,i2) を見つけるルーチンが必要になりました．固有ベクトルは，列ベクトルとして格納されている点に注意して print します．また for ループ内の演算結果は非表示ですから，そのためのセミコロンは不要です．

リスト 1 の内容のスクリプトを gp に読み込ませて，

データファイル（gp_dat）を作成しましょう．余計なメッセージが書き込まれないように，オプション ‘‘-q’’ を使います．また，~/.gprc の colors を ‘‘\\’’ でコメントアウトしてください．原因はわからないのですが，colors を設定すると，最初に制御コードが出力されてしまうようなのです．

```
$ gp -q < eigen.par > gp_dat
```

GP 自身もプロット命令を持っていますが，あまり機能が豊富ではありません．そこで，筆者愛用の gnuplot を用い図 1を作成しました．参考までに gnuplot のスクリプトをリスト 2 に示します．EPS ファイルが出来ますから，gv で見てください．

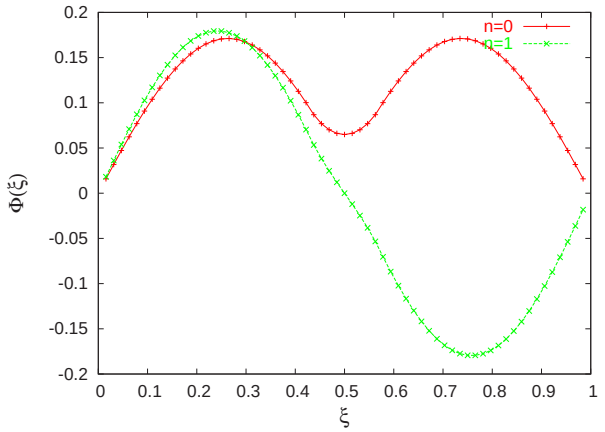

図 1　eigen.par 求まった波動関数を gnuplot で表示．

リスト 2 eigen.gp

```
set term postscript color eps "Helvetica" 20
set out "eigen.eps"
set xlabel "x" "Symbol,24"
set ylabel "F(x)" "Symbol,24"
plot "gp_dat"  t "n=0" w lp,\
     "gp_dat" using 1:3 t "n=1" w lp
```

任意精度計算の効能

前回から任意精度計算が可能なライブラリとツールをとりあげているわけですが，実際何の役に立つかといわれるとちょっと返答に困ります．GP の tutorial に大変判りやすい例が記されているので，それを紹介します．それは $\exp(\pi\sqrt{163})$ が整数か否かという問いかけです．20〜26 桁で計算するとと整数にみえますが，31 桁にすると整数でないことが判ります．

```
gp > \p 20
gp > exp(Pi*sqrt(163))
      262537412640768744.00
gp > \p 26
gp > exp(Pi*sqrt(163))
      262537412640768744.00000000
gp > \p 31
gp > exp(Pi*sqrt(163))
      262537412640768743.9999999999992
```

もちろん逆に整数であること証明する場合には何の役にも立たないかもしれませんが，実感できるという意味で素直に力を認めざるを得ません．実際，long double（20 桁）までしか精度を持っていない Octave や Scilab ではこの違いを明らかにすることができないのですから．

その他の任意精度計算ツール

ツールとして GP は完成度が高いですが，他にも大石進一先生ご推薦の calc[3]W[3]また神奈川工科大学の平山任弘先生が開発された高精度計算 c++ライブラリ mppack を用いた試作版ツール（これまた）calc[4]W[3]などは一見の価値があります．

参考文献

[1] PARI+GP の公式サイト
http://www.parigp-home.de/

[2] お近くの CTAN(TEX 関連ファイルのアーカイブ) ミラーサイトからどうぞ．例えば
ftp://ftp.iij.ad.jp/pub/TeX/CTAN/

[3] 柏木雅英先生の Calc ページ
http://www-kairo.csce.kyushu-u.ac.jp/
で紹介されている ~kashi/calc/homepage.ja.html でパッチをあてる版 1.24.7 は古いです．現在のメンテナンスサイトは
http://www.isthe.com/chongo/tech/comp/calc/
で，最新版は 2.11.4t2 となっています．

[4] 日本の高性能数値計算プロジェクト「Phase」のライブラリ
http://phase.etl.go.jp/phase/mppack/
から入手できます．ライブラリ mppack に関する丁寧なマニュアルがついていて自習用に最適です．前回紹介した apfloat よりも利用しやすいかも．