コンピュータと教育 25－4
（1993. 1. 22）

数式処理言語を用いた計算機教育*

牧野潔夫
工学院大学

計算機ハードウエアの進歩により以前には実用には今一歩だった数式処理言語の普及が最近著しい。数式処理言語には、科学技術用によく使われるFortranやUNIX世界の標準言語であるCにはない、多倍長の整数や小数、記号微積分などの機能がある。これらの機能を用いて工学部の2年生に数学の比較的簡単なアルゴリズムおよび計算機の入門、4年生に数式処理の特長をいかしたアルゴリズムとその実際のプログラムを教育した。扱った題材は初等性数論、直交多項式、代数方程式の数値解法、数値積分、ワークステーション、ユニックスである

Education of Computer Science with Symbolic Manipulation

Isao Makino
Kogauin University

Symbolic Algebraic Manipulation has many different functions form Fortarn, C. For examples, it has bignum, bigfloat, Symbolic derivation and integartion. Using these functions, we educated some algorithm in Mathematics and computer technology. More precisely, we educated elmentary number theory, orthogonal ploynomial, numerical solution of algebraic equation, numerical integration. Fathermore, we taught use of unix workstation.

*数式処理学会第一回研究報告会での発表に加筆

1 内容

工学部の少人数の学生(2年生、10人以内)に数学のアルゴリズムを教えながら計算機(特にワークステーション)の扱いを覚えさせることおよび4年生に数式処理特有の機能を使った実用になるプログラムの作成。

2 2年生の教育の概要

工学部でも学生に敬遠されがちな数学のアルゴリズムを扱うとき重要なことは学生が興味を持つ

1. 数学の素材(アルゴリズムの内容)
2. ハードウエア
3. ソフトウエア

の選択である。詳しく述べると

1。数学

計算アルゴリズムがよく解って計算機言語で記述しやすい部分多くある初等整数論、特に素数を扱った。

2. ソフトウエア

整数論では普通の数としてでてくる数百桁を何の工夫も用いないで扱える数式処理言語使用した。

3. ハードウエア

2. のソフトウエアを十分に動かせる能力(メモリ、cpu速度)を持ったハードウエアとしてワークステーション(hp apollo Domain 3500,3000,hp apollo 9000/700)を数台を用いた。

1.2.3. は互いに関連しており単独の項目だけでは決められない。

2.1 数学の内容

ここで扱う素数の性質は大きく分けると

1. 解析的な方法で扱う問題
2. 代数的な方法で扱う問題

に分けられる。(さらに代数幾何的方法もあるがここではここでは扱わない)

2.2 解析的方法

i. Riemann の ζ 関数と Euler 積

$$\sum_{n=1}^{\infty}\frac{1}{n^s}=\prod_{p:prime}(1-\frac{1}{p^s})^{-1}\quad(\Re s>1)$$

(素因数分解の一意性が必要)

ii. 素数の逆数の和

$$\sum_{p:prime}\frac{1}{p}$$

は収束するか?発散するか?
(ヒント $\sum_{n=1}^{\infty}\frac{1}{n}$ は発散し、$\sum_{n=1}^{\infty}\frac{1}{n^2}$ は収束する。また Riemann ζ 関数の Euler 積表示も用いる。)

iii. x までの素数の個数 $\pi(x)$ について。

$$c_1\frac{x}{\log x}<\pi(x)<c_2\frac{x}{\log x}$$

(c_1, c_2 は定数) は初等的に証明できる。

iv. その他興味ある式

$$\sum_{p\leq x}\frac{1}{p}=\log\log x+c+O(\frac{1}{\log x})$$

($c=0.2651...$) (i の答えになっている)

$$\sum_{p\leq x}\frac{\log p}{p}=\log x+O(1)$$

問題
$\sum_{n\leq x}\frac{1}{n}$ がはじめて(与えられた数)M を超える x は?
$\sum_{p:prime\leq x}\frac{1}{n}$ がはじめて(与えられた)M を超える x は?

v. 双子素数の問題

$p, p+2$ が共に素数になるとき $p(, p+2)$ を双子素数という。$\pi_2(x)=\#\{p\leq x \mid p, p+2$ は共に素数$\}$ と定める。
未解決問題

$$\lim_{x\to\infty}\pi_2(x)=+\infty??$$

しかし $\sum_p(\frac{1}{p}+\frac{1}{p+2})<\infty$ (ただし p は双子素数を動く) は解っている。

vi. Goldbach の問題

4 以上の任意の偶数は二つの素数の和で表せる。(未解決)

vii. Waring の問題

$k \geq 2$ とする。

ある数 r があって任意の数 n は r 個以内の k 乗数の和 ($n = \sum_{i=1}^{r} n_i{}^k$) で書き表せる。$G(k) = \min\{r\}$ とする。

ある数 s があって十分大きな任意の数 n は s 個以内の k 乗数の和 ($n = \sum_{i=1}^{s} n_i{}^k$) で書き表せる。$g(k) = \min\{s\}(\leq G(k))$ とする。

例

$k = 2$ のとき $g(k) = G(k) = 4$ である。

$k = 4$ のとき $g(4) = 19, G(4) = 16$ である。

2.3 代数的方法

i. 素数判定

Fermat の定理

$$p : prime, \quad (a,p) = 1 \quad \Longrightarrow$$

$$a^{p-1} \equiv 1 \pmod{p}$$

Euler の定理

$$p : prime, \quad (a,p) = 1 \quad \Longrightarrow$$

$$a^{\frac{p-1}{2}} \equiv (\frac{a}{p}) \pmod{p}$$

($(\frac{a}{p})$ は Legendre の記号)

Miller の定理

$p : prime, \quad (a,p) = 1, \quad p - 1 = 2^s d$

(ただし $(d,2) = 1$) とすると

$$a^d \equiv 1 \pmod{p}$$

または

$$a^{2^k} \equiv -1 \pmod{p}$$

for some k with $(0 \leq k \leq s-1)$

ii. 素因数分解

易しい方法をいくつか解説した。

3 4年生の教育

3.1 数式処理言語における数値積分

一般に数式処理言語は記号微積分が可能であるから不定積分が可能なら数値積分は必要ない。しかし不定積分の理論において強力である Rish のアルゴリズムはほとんどの数式処理ではその内容の十分の一ほどもインプリメントしておらず、また不定積分不可能な関数の定積分、例えば

$$\int_0^1 \frac{\sin(x)}{x} dx, \quad \int_0^1 t^{\sqrt{2}} \exp(-t) dt$$

などは数値積分以外に求める方法はない。
数値積分は高校の数学にも取り入れられており台形公式、シンプソンの公式が教科書にあらわれる。この二つは手計算が可能なように分点は等分、重みもすべて同じ値に取る計算法である。しかしここでは計算機を使うのであるから手計算ではかなり困難な、そのかわり誤差の少ない方法を考えてみたい。

3.2 数値積分概要

関数 $f(x)$ の定積分の値を求める方法でよく使われるものに $f(x)$ を定積分可能な関数で近似しこの関数を積分する方法である。$f(x)$ の近似関数が多項式のときは古くから考えられているが近年、近似代数的方法により近似関数が有理式のときも研究されている。

区間 $[a,b]$ での定積分を考える。$f(x)$ の補完多項式は $[a,b]$ の分割 $a \leq x_0 < x_1 < \ldots < x_n \leq b$ を用いると

$$f_n(x) = \sum_{i=0}^{n} f(x_i) p_i(x)$$

なる形をしている。ただし $p_i(x)$ は x の多項式である。従って $f(x)$ の $[a,b]$ での定積分は $f_n(x)$ の $[a,b]$ での定積分

$$\sum_{i=0}^{n} f(x_i)\int_a^b p_i(x)dx = \sum_{i=0}^{n} A_i f(x_i)$$

$$(A_i = \int_a^b p_i(x)dx)$$

で近似される。このときどのような $\{x_i\}$ と $\{A_i\}(p_i(x))$ をとれば精度が良くなるか? が問題となる。よく知られているように多項式をできるだけ少ない分点で近似する積分公式が Gauss の方法である。Gauss の方法を以下簡単に解説する。

$L^2([a,b],w(x)dx)$ $(w(x)\geq 0)$ の内積

$$(f,g) = \int_a^b f(x)g(x)w(x)dx$$

関し完備な基底 $1,x,x^2,...,x^n,...$ がとれる。この基底を正規直交化すると直交多項式

$$\phi_0(x),\phi_1(x),...,\phi_n(x),... \quad (\deg\phi_n(x)=n)$$

が得られる。とくに $a=-1,b=1,w(x)=1$ のとき $\phi_n(x)$ は Legendre の多項式 $P_n(x)$、$a=0,b=\infty,w(x)=\exp(-x)$ のとき $\phi_n(x)$ は Laguerre の多項式 $L_n(x)$、$a=-\infty,b=\infty,w(x)=\exp(-x^2)$ のとき $\phi_n(x)$ は Hermite の多項式 $H_n(x)$ である。$\phi_n(x)$ は次の三項の関係式を満足する。

定理 3.1

$$\phi_n(x)-(A_nx+B_n)\phi_{n-1}(x)+C_n\phi_{n-1}(x)=0$$

ただし

$$A_n=\frac{a_{n+1}}{a_n}, C_n=\frac{A_n}{A_{n-1}}, C_0=0\,(n=1,2,...)$$

で a_n は $\phi_n(x)$ の最高次 x^n の係数である。

さらに $\phi_n(x)=0$ の根はすべて単根であることが示される。これらを $x_1,x_2,...,x_n$ とする。$f(x)$ の補完多項式として

$$F(X)=\sum_{k=1}^{n} f(x_k)\frac{\phi_n(x)}{(x-x_k)\phi_n'(x)}$$

をとる。このとき

定理 3.2 $\deg F = n-1$ で

$$F(x_k)=f(x_k) \quad (k=1,2,...,n)$$

が成立する。この近似式を用いると

$$\int_a^b f(x)w(x)dx = \sum_{k=1}^{n}\lambda_{k,n}f(x_k)+E_n$$

で $2n-1$ 次の多項式 $f(x)$ に対し $E_n=0$ である。ただし

$$\lambda_{k,n}=\int_a^b \frac{\phi_n(x)}{(x-x_k)\phi_n'(x)}w(x)dx$$

である。この $\lambda_{k,n}$ は積分を用いずに次のようにあらわされる。

定理 3.3

$$\lambda_{k,n}=\frac{1}{\sum_{j=1}^{n}\phi_n^2(x_k)},$$

$$\lambda_{k,n}=-\frac{a_{n+1}}{a_n}\frac{1}{\phi_n'(x_k)\phi_{n+1}(x_k)}$$

従って $\lambda_{k,n}$
は積分を使わずに $x_k(k=1,2,...,n)$ および $\phi_j(x)(j=1,2,...,n)$ が解れば計算できる。さらに具体的に扱った直交多項式 $\phi_n(x)$ が Legnder,Laguerre,Hermite の多項式場合 $\lambda k,n$ はもっと易しい計算法があり, この方法を使えば、例えば Legnedre 多項式の場合 $\lambda_{k,n}$ は

$$\lambda_{k,n}=\frac{2(1-x_k^2)}{(nP_{n-1}(x_k))^2}$$

とあらわされる。

以上のことにより $f(x)w(x)$ の $[a,b]$ での定積分は次のようにして計算できる。

i. $f(x),n$ を与える。

ii. $\phi_k(x)(k=1,2,...,n)$ を定理 3.1 の漸化式で計算する。

iii. $\phi_n(x)=0$ の根 $x_j(j=1,2,...,n)$ を求める。

iv. $\lambda_{k,n}$ を定理 3.3 の式で求める。

v.

$$\sum_{k=1}^{n} \lambda_{k,n} f(x_k)$$

を計算する。
この値が

$$\int_a^b f(x)w(x)dx$$

の近似値である。

ここで問題となるのは iii の$\phi_n(x) = 0$ の根を求める部分である。代数方程式に限らず、方程式の全根の計算は難しく、数値計算の豊富な理論のある代表的分野の一つである。この問題を次のセクションで考える。

3.3 代数方程式の数値解法

代数方程式の全根を同時に求める優れた方法に Durand-Kerner 法がある。この方法は根が重根のときでも複素数のときでも使える良い方法である。しかしここでは解くべき方程式$\phi_n(x) = 0$ の根の性質が解っているため Graeffe の方法を用いた。この方法は根が全て実根で絶対値が異なるとき全根を同時に求める方法である。この方法の特長は、数式処理では標準の機能である多倍長の計算が基本で、この機能は現在の普通のプログラム言語では特別なことをしない限り持たない。また多項式 $p(x)$ に対し多項式 $(-1)^{\deg p} p(\sqrt{x})p(-\sqrt{x})$ を計算する部分があるがここも数式処理ではほとんど式の通りにプログラムできる。もう少し詳しく述べると REDUCE では

```
procedure ffsub(p,x);
begin
  return p*sub(x=-x,p)*((-1)^deg(p,x));
end;

procedure fsqx(funct,x);
   sub(x=sqrt(x),funct);
```

と簡単に書けるが式の扱いができなければ

$$p(x) = x^n + a_1 x^n + ... + a_k x^{n-k} + ... + a_n$$

$$(-1)^n p(\sqrt{x})p(-\sqrt{x})$$

$$= b_0 x^n + b_1 x^{n-1} + ... + b_k x^{n-k} + ... + b_n$$

とおいて $p(x)$ の係数 $a_k (k = 1, 2, ..., n)$ と $(-1)^{\deg p} p(x)p(-x)$ の係数 $b_k (k = 1, 2, .., n)$ の間の関係式

$$b_0 = 0, b_N = 0 \ (N > n), b_n = (-1)^n a_n^2,$$

$$b_i = (-1)^i a_i + 2 \sum_{j=0}^{i-1} (-1)^{j-1} a_j a_{2i-j}$$

をプログラムしなければならない。

この方法を使って$\phi_n(x) = 0$ を解くことができる。

4 ソフトウエア

4.1 使用言語

この教育の目的は数学のアルゴリズムを教えることが第一であって program 言語の教育は二次的なものである。そのため多倍長などの機能は予めからもっている言語で、かつまた言語文法の教育の負担をできる限り軽減するために、学生のよく用いている (知っている)pascal 言語に近い REDUCE および c 言語に近い risa/asir を使用した。

4.2 REDUCE,risa について

REDUCE は A.C.Hearn 氏が制作した最も古くからある数式処理言語の一つで最近普及している Mathematica と共にその動作するマシンの種類が多く、Cray から 8086 の P.C まで動く。risa は富士通の国際情報社会科学研究所で開発されている日本で作られた数少ない数式処理言語の一つである。現在の version は陰関数のグラフがかけ, 数式処理に必要な機能は最低限備えている。man power さえあればすぐにでも REDUCE 程度の機能は持つと思われ、さらには Macsyma,Mathematica,Maple 等に匹敵する処理系に成長する可能性もある。機能の拡充が他のソフトウエアとのリンクにより可能であるのが risa の一番の特長である。

現在数論関係者で話題になっている Free Soft である PARI GP の機能の一部がリンクされている。またなにか (バグ等) があったとき密に連絡のとれる人のいる処理系を使えることも重要なことである。

5 まとめ

i. 数式処理というプログラム言語の世界ではあまり知られていないソフトを用いて 2 年生と 4 年生を教育したが参考書が皆無の状態なので教材を全て自前で準備する必要があった。

ii. 一年間でできる内容は限られているので、機械の扱いは説明したくなかったが、unix マシンの使い方はとくに 2 年生で必要であった。

iii. 現在の高校までの数学では初等性数論では欠くべからざる合同式の話しが全くあらわない。そのため学生の興味のある素数判定などの話題になるまでかなり時間がかかり計算機使用法も行うと一年間では時間が足りない。

iv. 機能的にはかなりのものである数式処理も数値計算とくに小数の計算が遅く不満がある。

v. 近年の整数論の成果の一つである一般 Riemann 予想を仮定した素数判定法や楕円曲線、Hyper Elliptic curve を用いた素数判定法、Gauss,Jacobi 和を使った素数判定法の紹介とその計算量の非常に簡単な解説をした。学生は興味を示した。とくに計算量はアルゴリズムの改良でハードに頼らずに計算を早く行うことに関連するので興味を持ったと思われる。

vi. 最近の数式処理の世界では $f(x)$ の定積分は次の手順で行われることもある。
1. $f(x)$ の有理式近似 $\frac{F(x)}{G(x)}$ ($F(x), G(x)$ は多項式) を求める。
2. $G(x)$ を因数分解する。ここで $G(x) = 0$ を数値解法で解いたり、また近年盛んに研究されている近似代数計算の一部である近似因数分解を用いる。
3. $G(x)$ が無平方分解されれば $\frac{F(x)}{G(x)}$ の不定積分は簡単に求まりその定積分もすぐ計算される。
将来可能ならこの方法を 4 年生に教育してみたい。

参考文献

[1] 和田秀男　数の世界　岩波書店

[2] 高木貞治　初等整数論講義　共立出版

[3] 牧野潔夫　整数論の数式処理　サイエンチスト社

[4] 山内二郎、森口繁一、一松信　電子計算機のための数値計算法 I,II,III　倍風館

[5] 伏見康治、赤井逸　直交関数系 (増補版)　共立出版

付録　REDUCE での Gauss-Laguerre の積分プログラムと実行結果。

```
symbolic procedure factl(x);
begin
  integer a;
     a:=x;
  loop:
     if x=1 then return a;
     a:=a*(x-1);
     x:=x-1;
   goto loop;
end;

symbolic operator factl;

procedure ffsub(funct,x);
begin
  return funct*sub(x=-x,funct)*((-1)^deg(funct,x));
end;

procedure fsqx(funct,x);
   sub(x=sqrt(x),funct);

procedure ffnest(funct,x,k);
begin
  funct:=den(funct)*funct;
  for I:=1 : k do funct:=fsqx(ffsub(funct,x),x);
   return funct;
end;

procedure roots(funct,x,k);
begin
   scalar i,ll,n;
     n:=deg(funct,x);
     array xxl(n);
     ll:=reverse(coeff(funct,x));
     a:=first(ll);
     ll:=rest(ll);
     i:=1;
     loop:
       b:=first(ll);
       ll:=rest(ll);
```

```
         on bigfloat;on numval;
        write xxl(i):=(abs(b/a))^(1/k);
 i:=i+1;
          off numval;off bigfloat;
 if ll={} then return nil;
        a:=b;
      goto loop;
  end;

 procedure grf(funct,x,n);
   roots(ffnest(funct,x,n),x,2^n);

 procedure wkl(n,k);
   begin
     integer i;
     scalar l;
        l:=ln(n+1,x);
 grf(ln(n,x),x,k);
         array wgtkl(n);
         on bigfloat;on numval;
      for i:=1:n do <<wgtkl(i):=factl(n)**2*xxl(i)/(sub(x=xxl(i),l))^2;
                      write(wgtkl(i))>>;
         off numval;off bigfloat;
 end;

 procedure ln(n,x);
 begin
  scalar p0,p1,p,k;
  if n<0 then return "bad N";
  if n=0 then return 1;
  if n=1 then return 1-x;
  k:=2;
  p0:=1;
  p1:=1-x;
  loop:
    p:=(2*k-1-x)*p1-(k-1)**2*p0;
    p0:=p1;
    p1:=p;
    if k=n then return p;
    k:=k+1;
  goto loop;
 end;
```

```
procedure gauss10(funct,n,k);
begin
    integer i;
    scalar intg;
      wkl(n,k);
      integ:=0;
      i:=1;
      on bigfloat;on numval;
      loop:
        integ:=integ+wgtkl(i)*sub(x=xxl(i),funct);
        if i=n then return <<off numval;off bigfloat;integ>>;
        i:=i+1;
      goto loop;
end;

procedure gauss1(funct,a,n,k);
 gauss10(sub(x=x+a,funct)*exp(x),n,k);
end;

A:\LANG\REDUCE\SRC>reduce33
REDUCE 3.3, 15-Jan-88 ...
1: in "gaus_lag.red";
fx:=exp(-x);
       1
FX := ----
       X
      E
gx:=exp(-x)*sin(x);
      SIN(X)
GX := --------
        X
       E
hx:=exp(-x)*sqrt(x);
      SQRT(X)
HX := ---------
        X
       E
on bigfloat;
precision 30;
30
off bigfloat;
gauss1(fx,0,4,7);
0.999 99999 99999 99999 99999 99999 98 %Anser =1
```

```
gaussl(gx,0,6,8);
*** ARRAY XXL REDEFINED
XXL(1) := 15.98 28739 80601 70178 25457 91567 4
XXL(2) := 9.837 46741 83825 89917 71554 70299 4
XXL(3) := 5.775 14356 91045 10501 83983 03694 3
XXL(4) := 2.992 73632 60593 14077 69132 52845 1
XXL(5) := 1.188 93210 16726 23030 74315 09219 4
XXL(6) := 0.222 84660 41792 60689 46435 48267 87
*** ARRAY WGTKL REDEFINED
0.000 00089 85479 06429 62123 88252 92052 814
0.000 26101 72028 14932 05947 92428 60001
0.010 39919 74531 49074 89891 33028 47
0.113 37338 20740 44975 73870 61850 98
0.417 00083 07721 20994 11337 75661 94
0.458 96467 39499 63593 56828 48777 09
0.500 04947 47976 75038 94438 12721 57 %Anser =1/2
gaussl(hx,0,6,8);
0.893 29552 22343 74897 52609 50305 07 %Anser
ws*ws*4;
3.191 90756 01759 38307 65388 29215 5  %=pi
end;
2: bye;
```