最良近似式計算システム

浜田穂積（日立製作所中央研究所）

1．はじめに

1変数の解析関数$f(x)$の，実数値の一つの区間における最良近似式$g(x)$を計算する問題について述べる。近似式を定める区間を近似区間と呼び，プログラムの見やすさの点から次の通りとする。

(1)　　$q-r \le x \le q+r \qquad (r>0)$

qは区間の中点，rは半巾である。近似式$g(x)$の形を

(2p)　　$g(x)=c_1+c_2x+\cdots+c_nx^{n-1}$

あるいは

(2c)　　$g(x)=\cfrac{1}{c_1+\cfrac{x}{c_2+\cdots+\cfrac{x}{c_n}}}$

であるとして，実係数$c_i\ (i=1,2,\cdots,n)$を求めるのがここでの目的である。ここでは多項式形式と連分数形式の近似式の計算法に強い類似性があることを示すため，両形式の対応する式にそれぞれpあるいはcを添えて示す。単に番号のみで参照する場合はそれらをまとめて区別せず参照する場合である。また自由に定めうる係数の個数nを自由度と呼ぶ。

最良近似式を求める対象としての関数$f(x)$を解析関数と規定したが，実際にはもっと条件をゆるめることができる。しかし大部分の場合解析関数で十分である。もし複数の区間でそれぞれ解析関数をつぎ合せたものをまとめて近似しようとするならばそれは望ましくない。それぞれの区間での近似式を求めるほうがよい。

2．最良近似式の計算法

最良近似式であるための条件から作られる方程式から直接計算することは困難なので，次に述べるよく知られた収束計算による。

誤差の評価式を$E(x)$とする。次の2式を満たす$n+1$個の点x_jを偏差点という。($x_0=q+r>x_1>\cdots>x_{n-1}>x_n=q-r$）最良近似式では次が成り立つ。

(3)　　$E(x_j)=(-1)^j\varepsilon \qquad (j=0,1,2,\cdots,n)$

(4)　　$\left.\dfrac{dE(x)}{dx}\right|_{x=x_j}=0 \qquad (j=1,2,\cdots,n-1)$

(3),(4)が成り立つように次の手順で解く。

(i)　c_i, x_jの初期値を適当に定める。

(ii)　x_jを固定して，(3)を満たすようにc_iを定める。

(iii)　c_iを固定して，(4)を満たすようにx_jを定める。

(iv)　$E(x_j)$のバラツキを調べ，収束したとみなせないとき(ii)へ戻る。

なお誤差の評価式は，絶対誤差に関する近似のとき

(5)　　$E(x)=g(x)-f(x)$

相対誤差に関する近似のとき

(6)　　$E(x)=\{g(x)-f(x)\}/f(x)$

とする。

3. 一般の関数，偶関数，奇関数および絶対誤差，相対誤差

$f(x)$が偶関数あるいは奇関数であるとき，(1)における$g=0$とすれば，その性質を用いて自由度が小でも比較的高精度の近似式を得ることができる。

偶関数の絶対誤差に関する近似ではx^2を改めてxとおき，区間を$0 \le x < r^2$とおけば，一般の関数の場合に帰着できる。

相対誤差に関する近似の場合は，$f(x)$が$O(x^m)$であるとするとき$\bar{f}(x)=f(x)/x^m$に関する最良近似式$\bar{g}(x)$を求め，$g(x)=\bar{g}(x)\cdot x^m$を求める$f(x)$の最良近似式とする。したがって奇関数に関する場合は偶関数に帰着でき，偶関数は上と同じ方法で一般の関数の場合に帰着できるので，表題の組合せ6通りは，表1の本質的には3通りの場合に分類できる。

表1．本質的な場合

評価＼式の形	一般の関数	偶関数	奇関数
絶対誤差	a	←	o
相対誤差	r	←	←

この表で○は本質的なもの，←は左の欄に帰着できることを示す。○の中に示したのはそれぞれの場合を示すシンボルである。これらそれぞれに多項式形と連分数形の近似式があるので，略記シンボル

pr, pa, po, cr, ca, co

の6通りのプログラムでほぼ間にあうことになる。奇関数の場合，偏差点は$2n+2$個あるが，対称性から本質的な半分の$n+1$個（正の方）のみを考慮する。またこのとき(4)のjは$1 \sim n$である。

ここで未知数と式の個数を整理しておく。

- 一般の関数
  - 未知数：$c_i\ (i=1,2,\cdots,n)$，$x_j\ (j=1,2,\cdots,n-1)$，εの$2n$個
  - 式　　：(3)の$n+1$個，(4)の$j=1,2,\cdots,n-1$の$n-1$個で$2n$個
- 奇関数
  - 未知数：$c_i\ (i=1,2,\cdots,n)$，$x_j\ (j=1,2,\cdots,n)$，εの$2n+1$個
  - 式　　：(3)の$n+1$個，(4)の$j=1,2,\cdots,n$のn個で$2n+1$個

4. 計算法

一般の関数の場合，$f(x)$を次の漸化式で表わす。

(7)　$y_n = f_r(x)$

(8p)　$y_{i-1} = (a_i + x y_i)/b_i$

(8c)　$y_{i-1} = b_i/(a_i + x y_i)$

$\Big\}\ (i=n,\cdots,2,1)$，$b_i \neq 0$

(9)　$f(x) = y_0$

ここで剰余関数$f_r(x)$は，区間(1)でおだやかな関数となるように，a_i, b_iを選ぶ。具体的には$y_0(=f(x))$から出発して$i=1,2,\cdots,n$の順に

(10p)　$xy_i = b_i y_{i-1} - a_i$　から　$b_i y_{i-1} - a_i$

(10c)　$xy_i = b_i/y_{i-1} - a_i$　から　$b_i/y_{i-1} - a_i$

が$O(x)$となるように，かつb_iは整数，a_iもなるべく整数になるように選ぶ。奇関数の場合は(8)の右辺にxを掛けたものである次とする。(7),(9)は同じである。

(11p)　$y_{i-1} = x(a_i + x y_i)/b_i$

(11c)　$y_{i-1} = x b_i/(a_i + x y_i)$

またa_i, b_iについては次が$O(x^2)$となるように選ぶ。

(12p)　$b_i y_i/x - a_i$

(12c)　$b_i x/y_i - a_i$

$f_r(x)$の計算法はそれぞれの展開形の延長であってもよいし，他の方法であってもよい。$f_r(x)$を精度よく計算できるかどうかが，この計算の鍵である。

$g(x)$は$f(x)$における係数a_iに対する補正量d_iを導入して次の通りとする。一般の関数の場合

(13)　$z_n = 0$

(14p)　$z_{i-1} = (a_i + d_i + x z_i)/b_i$

(14c)　$z_{i-1} = b_i/(a_i + d_i + x z_i)$　$\Big\}\ (i = n, \cdots, 2, 1)$

(15)　$g(x) = z_0$

である。奇関数の場合は(11)と同様である。

$e(x) = g(x) - f(x)$とするとき同様に次の通りである。

(16)　$e_n = -f_r(x)$

(17p)　$e_{i-1} = (d_i + x e_i)/b_i$

(17c)　$e_{i-1} = -y_{i-1} z_{i-1}(d_i + x e_i)/b_i$　$\Big\}\ (i = n, \cdots, 2, 1)$

(18)　$e(x) = e_0$

奇関数のときは同様に(17)の右辺にxを掛ける。$E(x)$は(5),(6)より，絶対誤差のとき$e(x)$，相対誤差のとき$e(x)/f(x)$である。

計算手順(ii)でd_iが正しくないため(3)を満たさないとき，d_iの修正量をΔd_iとして，$d_i - \Delta d_i$とすると(3)を満たすと考えた式を作る。すなわち

(19)　$E(x_{j-1}, d_1 - \Delta d_1, \cdots, d_i - \Delta d_i, \cdots) + E(x_j, d_1 - \Delta d_1, \cdots, d_i - \Delta d_i, \cdots) = 0$

である。x_jを固定して，Δd_iが微小量とすると次の通りとなる。絶対誤差の場合

(20)　$\left\{\frac{\partial e(x_{j-1})}{\partial d_1} + \frac{\partial e(x_j)}{\partial d_1}\right\}\Delta d_1 + \cdots + \left\{\frac{\partial e(x_{j-1})}{\partial d_n} + \frac{\partial e(x_j)}{\partial d_n}\right\}\Delta d_n = e(x_{j-1}) + e(x_j)$

ここで重要なのは$\partial e(x)/\partial d_i$であり，これを$h_i$とする。相対誤差の場合$f(x)$には$d_i$を含まないので$\partial e(x)/\partial d_i$を$f(x)$で割ったものを$h_i$とする。このとき

(21p)　$h_1 = 1/b_1$

(21c)　$h_1 = -z_0^2/b_1$

(22p)　$h_i = x/b_i \cdot h_{i-1}$

(22c)　$h_i = -z_{i-1}^2 x/b_i \cdot h_{i-1}$　$\Big\}\ (i = 1, 2, \cdots, n)$

により計算する。奇関数の場合も同様である。

直接$a_i + d_i$にあたるものを求めないでd_iを求めるのは，$g(x) - f(x)$の計算による桁落ちを防ぐためで，戸田英雄教授等のアイデアに基づいている。

(20)は未知数Δd_iに関するn個の式からなる連立方程式であり，Δd_iを解いてd_iを改める。以上のようにして解いたd_iを用いて最終的に$g(x)$を(13),(14),(15)から(2)の形に変形する。これは次の通りである。

(23p)　$c_i = (a_i + d_i)/(b_i \cdots b_2 b_1)$

(23c)　$c_i = (a_i + d_i)/(b_i/\cdots/(b_2/b_1)\cdots)$

5. 変数変換

$E(x)$が，一般の関数の場合はn次の，奇関数の場合は$2n+1$次のチェビシェフ関数を一次変換したものに近い形をしていることが経験的に知られているので，

x を次式により p に置き換えたものを考える。一般の関数の場合

(24)　$x = r\cos(\frac{p}{n}\pi) + q$　　$(0 \le p \le n)$

奇関数の場合

(25)　$x = r\cos(\frac{p}{2n+1}\pi)$　　$(0 \le p \le 2n+1)$

このとき x_j に対応する p_j は j に近いということになる。図1には，e^x を $0 \le x \le \ln 2$ の範囲で多項式近似したものの，x から p への射影の結果の $E(p)$ のカーブを示す。x から p に変数を変換する利点は次の2つである。

- $E(p)$ のカーブが p_j を中心にその付近で極めて対称に近いので，計算手順(iii)における $E(p)$ の極大点を求めるのに好都合である。
- 同じ関数の同じ範囲における近似式を，種々の n について求めるとき，n の小なる値(ex. 1)から順に1ずつ増して求めるのが計算しやすいが，p_j の j からの偏りの傾向は n の変化で大きくは変らないため，p_j の初期値を推測しやすい。

　これらの利点を利用して，p_j の初期値は，求めるべき p_j を j の小なるものと大なるものの2グループに分け，小なるものについては $p_j = p_j$(n が1つ小のものの)，大なるものについては，$p_j = p_{j-1}$(n が1つ小のものの)$+1$ とすることでうまくゆく。
また $E(p)$ の極大点をとる p_j の値の計算は，これまでの p_j を中心とする5点での2段の中心差分により，ニュートン法を1回適用するだけで十分よい結果を得た。
すなわち $E'(p)=0$ の解の近似値として前回の p_j をとり，

(26)　$p_j - E'(p_j)/E''(p_j)$

を新しい p_j とする。ここに $E'(p_j)$, $E''(p_j)$ は次の5点の中心差分による。

(27)　$E'(p_j) = \{E(p_j+2h) - 8E(p_j+h) + 8E(p_j-h) - E(p_j-2h)\}/(12h)$

(28)　$E''(p_j) = \{E(p_j+2h) - 16E(p_j+h) + 30E(p_j) - 16E(p_j-h) + E(p_j-2h)\}/(12h^2)$

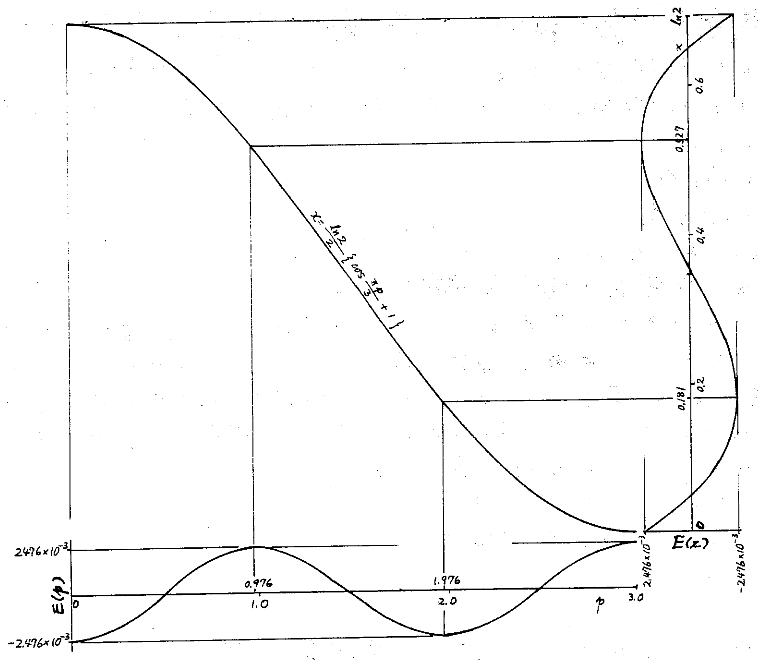

図1. 独立変数の変換

6．計算結果の有意桁数の決定

計算手順(v)での，収束したかどうかの判定は次による。$\varepsilon_j = E(p_j)$ として，ε_0 の符号をs（$\varepsilon_0>0$ のとき $s=1$，$\varepsilon_0<0$ のとき $s=-1$）とする。

(29) $\alpha_j = (-1)^j s \varepsilon_j$

として，α_j の最大値を ε_{max}，最小値を ε_{min} とする。このとき収束度 κ を次によって定義する。

(30) $\kappa = (\varepsilon_{max} - \varepsilon_{min})/\varepsilon_{max}$

ε_{max} はこの計算の結果として重要な最大誤差である。κ は実用的には 10^{-2} 以下程度で十分であるが，プログラムでは 10^{-10} としてある。ただし，性質の悪い場合に無限に繰返してはいけないので，たかだか9回までしか繰返さないことにした。これらの収束判定条件は絶対的なものではない。

次に，計算によって得られた d_i がどれくらいに信用できるか，いいかえると，(14)を展開して(2)の形にした場合の c_i として何桁有効かを定める方法を，2つの観点から述べる。

・d_i の誤差が最大誤差に与える影響

d_i に微小な変化を与えたときに生ずる $E(x)$ の変化のうち，偏差点におけるものが最も重要であるが，これは正に $\partial E(x)/\partial d_i$ であり，これは(21),(22)で求められるものである。そこで $E(x)$ が ε_{max} 程度に変化する η_i，すなわち

(31) $\eta_i \max_j \left| \frac{\partial E(x_j)}{\partial d_i} \right| = \varepsilon_{max}$

である η_i を求める。

・計算精度の d_i に与える影響

最良近似式の計算が単精度で行なわれたか，倍精度で行なわれたかの違いは，これまでの記述に関しては直接的には反映されていない。これの反映される指標を導入する必要がある。d_i に，異なる値として区別できる最小の変化 β_i を与えて $d_i+\beta_i$ としたものによって Δd_k を求める。理想的には

(32) $\Delta d_i = -\beta_i$,　$\Delta d_k = 0$ $(k \neq i)$

となるはずであるが，計算が有限精度で行なわれるためそうならない。ここで γ_{ik} を，d_i を $d_i+\beta_i$ としたときの Δd_k の値とする。そして

(33) $\zeta_i = \sum_{k=1}^{n} |\gamma_{ik} + \beta_i \delta_{ik}|$　$(\delta_{ii}=1,\ \delta_{ik}=0 : i \neq k)$

を求める。ζ_i は計算が有限精度で行なわれるために生じる誤差のうち，d_i に関係するものの総和である。η_i, ζ_i は結果の出力にあたって，d_i に関するものを直接出力するのでなく，c_i の値に換算して出力する方が便利である。また10進での桁数で考えやすいように，κ も含めて $-\log_{10}\kappa$, $-\log_{10}\eta_i$, $-\log_{10}\zeta_i$ で出力するようにプログラムはなっている。計算結果の取捨は，たとえば，

(34) $-\log_{10}\eta_i + 4 < -\log_{10}\zeta_i$

がすべての i について成り立てば採用できると考えてよいであろう。

7．プログラム

以上述べた点を反映した最良近似式計算プログラムの構造は次ページの通りである。これは先に述べた6種についてまったく変りはない。内部の細かい処理については異なるものもあり，同じものもある。その点も示してある。

メインプログラム　n：自由度，　ps：ピボット列

<u>procedure</u> decompose(A)
行列AのLU分解を行なう。
(共通)

<u>procedur</u> solve(A, B, C)
LU分解された行列AとベクトルBを右辺とする連立方程式を解いて解ベクトルをCに入れて返す。
(共通)

<u>procedure</u> minmax(r, q, a, b, fr)　maxerr：ε_{max}，conv：κ, d

<u>function</u> er(x)
$E(x)$の計算
(すべて異る)

<u>procedure</u> deriv(x, w)
$\dfrac{\partial E(x)}{\partial d_i}$ を w_i に計算して設定する。
(すべて異る)

<u>function</u> proj(p)
(24)あるいは(25)による p から x への射影を行なう。
(r, a と o)

<u>procedure</u> initpd

<u>procedure</u> coeff
方程式の係数計算を行なう。
(すべて異る)

- p_jの初期値設定（計算手順(i)）
- $E((p_{j-1}+p_j)/2)=0$ となる方程式の作成。d_iの初期値計算。(共通)

<u>procedure</u> refined
d_iの修正。（計算手順(ii)）
(共通)

<u>procedure</u> refinep
p_jの修正。（計算手順(iii)）
(共通)

<u>procedure</u> check
収束度 κ と ε_{max}，$E(p_j)/\varepsilon_{max}$ の計算
(共通)

<u>procedure</u> final
η_i，ζ_iの計算。c_i等の印刷。
(p と c)

図4参照

(出力を除いて p と c で異る)

<u>function</u> fr(x)
剰余関数値 fr(x) の計算

図3参照

図2．プログラムの構造

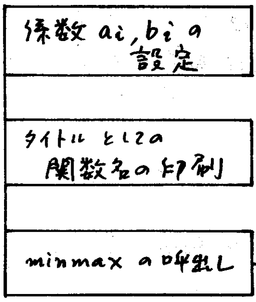

図3．メインプログラム

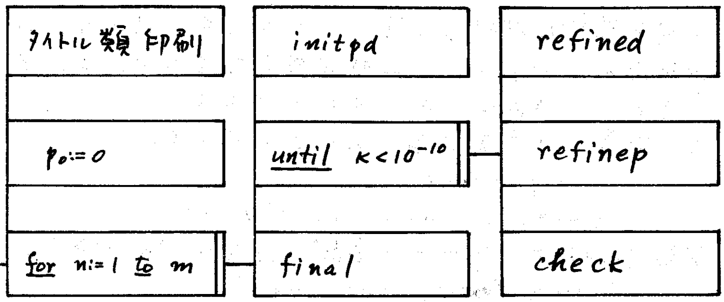

図4．計算の主要部

このプログラムはPascalで記述した。この種のプログラムをPascalで記述する例はあまり見聞しないが，十分その威力を発揮したと思う。残念なのは，十分よく最適化された目的プログラムを出すコンパイラがないことである。しかし，このプログラムに関するかぎり，十分計算は速い（HITAC M200Hで）ので，この欠点はまったく感じない。

Pascalは文字の編集が記述しやすいので，それを用いて一見高精度計算を行なう部分も用意した。というのは ε_{max} とくらべ d_i の有効桁数はそれほど大きくないので，a_i+d_i は相当高精度でも，a_i, d_i はそれほどでなく，十分倍精度で表わし得る。これを用いて最終結果だけを多倍長で計算することが可能であるからである。minmax 部分の6つのプログラムの平均は約80行である。

8．計算例

下に指数関数 e^x の多項式による絶対誤差に関する最良近似式の係数の計算結果を示す。近似範囲は $0 \le x \le \log_e 2$ である。

計算日付 83-07-18　近似式の形 poly

p_i	e_i/e_{max}	η_i（繰返し回数）	ζ_i（κ）
0.000000	-1.000	1	16.56
1.000000	1.000	0.30	17.16
0.000000	-1.000	3	15.19
0.963353	1.000	1.37	16.50
2.000000	-1.000	1.21	17.94
0.000000	-1.000	3	15.17
0.976192	1.000	2.61	17.43
1.976124	-1.000	2.45	16.33
3.000000	1.000	2.29	16.18
0.000000	-1.000	3	10.17
0.996191	1.000	11.96	22.24
1.992855	-1.000	11.80	19.92
2.990400	1.000	11.64	18.39
3.989120	-1.000	11.48	17.29
4.989159	1.000	11.32	16.48
5.990498	-1.000	11.16	15.94
6.992967	1.000	11.00	15.65
7.996264	-1.000	10.84	15.66
9.000000	1.000	10.68	16.05

関数名→exp(x)　r=0.3465736 ←近似区間の半巾

n	MAE	i	c[i]
1	5.000E-01	1	1.50000
2	4.304E-02	1	0.956964
		2	1.442695
3	2.476E-03	1	1.0024761
		2	0.9392620
		3	0.7159932
9	1.102E-12	1	1.0000000000011018
		2	0.9999999997444406
		3	0.5000000097546248
		4	0.1666665234470971
		5	0.0416677181961408
		6	0.0083290098532745
		7	0.0013992721150460
		8	0.000184047592085
		9	0.000035203680625

図5．計算結果

この図で罫線は印刷後手書きで入れたものである。左半分はチェック用で，右半分が係数表とできるよう体裁を整えたものである。左2列は偏差点にかかわるもの，その右は係数にかかわるものである。

この計算に必要な準備は次のものである。

$$r = q = \ln 2/2$$

$$a_i = 1 \quad ; \quad b_1 = 1 \ , \ b_i = i-1 \ (i>1)$$

関数 $fr(x)$ は，テーラー展開の延長が便利である。これを fexp とすると

```
function fexp(x:real):real;
  const l=15;  var i:1..l; s:real;
begin  s:=0;
  for i:=l downto 1 do  s:=(s*x+1)/(n+i-1);
  fexp:=s end
```

とすればよい。

9．おわりに

最良近似式を計算するプログラムを，多項式，連分数の形式それぞれ3種ずつ計6種作成した。これで通常必要となるものはほぼすべて計算可能である。これらのプログラムの構成はすべて同じで，非常に類似性の強い処理内容となっている。変数変換を用いることにより，誤差関数のふるまいが素直になってトラブルの発生を防いでいるといえよう。結果の有意桁の決定も計算精度の影響を考慮に入れて完全なものになったといえよう。プログラミング言語で使用される初等外部関数のための係数を計算したが，いずれも楽々と計算できた。計算方式を変えていろいろ試みることも何の苦もなく行なえるようになった。

なお本プログラム作成に用いたPascalコンパイラは，東京工業大学総合情報処理センターの御好意により使用させていただいているものである。同センターの前野年紀助教授に感謝いたします。

参考文献

1). Forsythe, G.E. and Moler, C.B.（渋谷政昭，田辺国士訳）：計算機のための線形計算の基礎，Prentice-Hall（培風館），(1967)

2). 浜田穂積：有理式近似および連分数近似の最良化について，情報処理，vol.19, no.11, pp1065-1071 (1978)

3). 一松信：初等関数の数値計算，教育出版 (1974)

付録. 計算例で示したもののプログラム

```
program minmax(output);
const m=9; ln2=0.6931471805599453O9;
type Re=real; fd=1..m;
  afdr=array[fd] of Re; afdfdr=array[fd] of afdr;
var n:fd; a,b:afdr; ps:array[fd] of fd;
procedure decompose(var a:afdfdr);
  var i,j,k:fd; t:Re;
begin for i:=1 to n do ps[i]:=i;
  for k:=1 to n-1 do
  begin i:=k; t:=abs(a[k,k]); for j:=k+1 to n do
    if t<abs(a[j,k]) then begin i:=j; t:=abs(a[j,k]) end;
    if i<>k then
    begin j:=ps[i]; ps[i]:=ps[k]; ps[k]:=j; for j:=1 to n do
      begin t:=a[i,j]; a[i,j]:=a[k,j]; a[k,j]:=t end end;
    for i:=k+1 to n do
    begin t:=-a[i,k]/a[k,k]; a[i,k]:=-t; for j:=k+1 to n do
      a[i,j]:=a[k,j]*t+a[i,j] end end end;
procedure solve(var a:afdfdr;var b,c:afdr);
  var i,j:fd; s:Re;
begin for i:=1 to n do
  begin s:=b[ps[i]]; for j:=1 to i-1 do s:=s-a[i,j]*c[j];
    c[i]:=s end;
  for i:=n downto 1 do
  begin s:=c[i]; for j:=n downto i+1 do s:=s-a[i,j]*c[j];
    c[i]:=s/a[i,i] end end;
procedure minmaxpa(r,q:Re;a,b:afdr;function fr(x:Re):Re);
  type dp=0..m; adpr=array[dp] of Re;
  var j:dp; k:fd; l:0..9; maxerr,conv:Re; d:afdr; e,p:adpr;
    dd:packed array[1..8] of char;
  function er(x:Re):Re;
    var i:fd; e:Re;
  begin e:=-fr(x); for i:=n downto 1 do e:=(e*x+d[i])/b[i];
    er:=e end;
  procedure deriv(x:Re;var w:afdr);
    var i:fd; s:Re;
  begin s:=1; for i:=1 to n do
    begin s:=s/b[i]; w[i]:=s; s:=s*x end end;
  function proj(p:Re):Re;
  begin proj:=r*sino(2-p/(0.25*n))+q end;
  procedure initpd;
    var j:dp; g:afdr; c:afdfdr;
    procedure coeff(x:Re;var w:afdr;var s:Re);
      var i:fd;
    begin s:=1; for i:=1 to n do
      begin s:=s/b[i]; w[i]:=s; s:=s*x end;
      s:=s*fr(x) end;
  begin for j:=n-1 downto n div 2 do p[j+1]:=p[j]+1;
    for j:=1 to n do coeff(proj((p[j-1]+p[j])/2),c[j],g[j]);
    decompose(c); solve(c,g,d) end;
  procedure refined;
    var i:fd; j:dp; u,v,x:Re; f,g,t:afdr; c:afdfdr;
  begin u:=er(r+q); deriv(r+q,g); for j:=1 to n do
    begin v:=u; t:=g; x:=proj(p[j]); u:=er(x); deriv(x,g);
      f[j]:=u+v; for i:=1 to n do c[j,i]:=g[i]+t[i] end;
```

```
    decompose(c); solve(c,f,g);
    for i:=1 to n do d[i]:=d[i]-g[i] end;
  procedure refinep(var q:Re);
    const h=0.01; var v,w,x,y,z:Re;
  begin v:=er(proj(q-h-h)); w:=er(proj(q-h));
    x:=er(proj(q)); y:=er(proj(q+h)); z:=er(proj(q+h+h));
    q:=q-h*((z-v)-8*(y-w))/(z+v-16*(y+w)+30*x) end;
  procedure check;
    var j:dp; s,t,u:Re;
  begin t:=er(r+q); e[0]:=t; s:=t/abs(t); t:=t*s;
    maxerr:=t; u:=t; for j:=1 to n do
    begin t:=er(proj(p[j])); e[j]:=t; s:=-s; t:=t*s;
      if maxerr<t then maxerr:=t; if u>t then u:=t end;
    conv:=(maxerr-u)/maxerr;
    for j:=0 to n do e[j]:=e[j]/maxerr end;
  procedure final;
    var i,k:fd; j:dp; ac,s,u,v:Re; f,g,t:afdr; w:afdfdr;
  begin deriv(r+q,g); for i:=1 to n do f[i]:=abs(g[i]);
    for j:=1 to n do
    begin t:=g; deriv(proj(p[j]),g); for i:=1 to n do
      begin if f[i]<abs(g[i]) then f[i]:=abs(g[i]);
        w[j,i]:=g[i]+t[i] end end;
    decompose(w); for i:=1 to n do
    begin f[i]:=f[i]/maxerr; s:=d[i]; d[i]:=s+eps(s);
      u:=er(r+q); for j:=1 to n do
      begin v:=u; u:=er(proj(p[j])); g[j]:=u+v end;
      d[i]:=s; solve(w,g,g); g[i]:=g[i]-eps(s);
      if i=1 then for j:=1 to n do t[j]:=abs(g[j]) else
      for j:=1 to n do t[j]:=t[j]+abs(g[j]) end;
    if conv=0 then s:=99.99 else s:=-log(conv);
    writeln(output,p[0]:10:6,e[0]:7:3,l:6,s:6:2);
    u:=1; for i:=1 to n do
    begin u:=u*abs(b[i]); s:=log(f[i]*u);
      if t[i]=0 then v:=99.99 else v:=log(u/t[i]);
      write(output,p[i]:10:6,e[i]:7:3,s:6:2,v:6:2);
      if i=1 then write(output,n:3,maxerr:10)
                else write(output,' ':13);
      ac:=a[i]+d[i]; for k:=i downto 1 do ac:=ac/b[k];
      writeln(output,i:3,ac:trunc(s)+10:trunc(s)+5) end end;
begin writeln(output,'(x)  r=',r:9:7);
  date(dd); writeln(output,dd:10,'poly':7);
  writeln(output,'n':32,'MAE':7,'i':6,'c[i]':9);
  p[0]:=0; for k:=1 to m do
  begin n:=k; initpd; l:=0;
    repeat refined; for j:=1 to n-1 do refinep(p[j]);
      check; l:=l+1 until (conv<1e-10)or(l=9);
    final end end;
function fexp(x:Re):Re;
  const l=15; var i:1..l; s:Re;
begin s:=0; for i:=l downto 1 do s:=(s*x+1)/(n+i-1);
  fexp:=s end;
begin for n:=1 to m do a[n]:=1;
  b[1]:=1; for n:=2 to m do b[n]:=n-1;
  page(output); write(output,'exp':43);
  minmaxpa(ln2/2,ln2/2,a,b,fexp) end.
```