Lagrange 補間と Spline 補間の計算とグラフ表示
Ｊによる「原理的」計算とプログラムの作成

西川　利男

先月の JAPLA 例会で志村正人氏より Spline 曲線についての紹介があった。
これを受けて、Lagrange 補間と Spline 補間とについて、その原理に基づいたごく素朴な説明とプログラムの作成を報告する。私にとっては、あらためてこのテーマを勉強する機会にもなり、さらにグラフ表示も含めてＪの環境がいかに有効であるかを実感することになった。

１．テスト・データ

一般的な数式による説明よりも、具体的な数値で示した方がずっとわかり易い。ここでは、戸田、小野の教科書[1]を元に私なりに説明し、プログラムとして示す。

次のような実験結果が得られたとする。（数値例は同じ）

X	-3	-1	0	2	5
Y	-18	-8	10	50	26

２．Lagrange 補間とそのプログラム

上のような５つの値(X, Y)が実験により測定されたとき、その途中の値がどのような値になるか予測したい。

このような補間を高次多項式を使って行うのが Lagrange の補間である。数学書ではこの後 Lagrange の補間多項式などが出てくるが、ここではそんなものは必要ない。

２つの点の補間なら１次式、３つの点なら２次式で充分である。今は５つの点なので４次の多項式で補間する。

ここで、4 次多項式関数を次のように仮定する。

$$f(x) = ax^4 + bx^3 + cx^2 + dx + e \quad \cdots\cdots\cdots\cdots \quad (1)$$

上の測定結果から

$$a(-3)^4 + b(-3)^3 + c(-3)^2 + d(-3) + e = -18$$
$$a(-1)^4 + b(-1)^3 + c(-1)^2 + d(-1) + e = -8$$
$$\cdots \quad \cdots$$
$$a\,5^4 + b\,5^3 + c\,5^3 + d\,5 + e = 26$$

これを行列演算の形で書けば次のようになる。

$$\begin{pmatrix} (-3)^4 & (-3)^3 & (-3)^2 & (-3)^1 & (-3)^0 \\ (-1)^4 & (-1)^3 & (-1)^2 & (-1)^1 & (-1)^0 \\ & \cdots & \cdots & & \\ 5^4 & 5^3 & 5^2 & 5^1 & 5^0 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \\ e \end{pmatrix} = \begin{pmatrix} -18 \\ -8 \\ 10 \\ 50 \\ 26 \end{pmatrix}$$

すなわち、この連立方程式を解けば、４次多項式関数(1)式の係数 a, b, c, d, e が求まる。したがって、(1)式により任意の値についての補間した値が得られる。

[1] 戸田英雄、小野令美「入門数値計算」p.35-43, オーム社 (1983).

これにより、そのまま直ちにＪでプログラミングすることができる。
実験結果を次のようにする。

```
   XL =: _3 _1 0 2 5
   XL
_3 _1 0 2 5
   YL =: _18 _8 10 50 26
   YL
_18 _8 10 50 26

   NL =: |. i. #XL
   NL
4 3 2 1 0
```

係数行列ALは次のようになる。

```
   AL =: |: XL ^/ NL
   AL
 81   1 0 16 625
_27 _1 0  8 125
  9  1 0  4  25
 _3 _1 0  2   5
  1  1 1  1   1
```

これを使って、連立方程式の解は、Ｊのプリミティブ %. により一発で求められる。
そして、この解の値がLagrangeの補間係数FLとなる。

```
   FL =: YL %. AL
   FL
2.33333 _39 52.8667 10.3333 _0.533333
```

以上の過程をプログラムlagrとして定義した。

```
lagr =: 3 : 0"(1 0)
:
'X Y' =. x.
N =. |. i. #X
A =. |: (|. i. #X) (^~/) X
F =. Y %. A
XX =. N ^~ y.
+/ F * XX
)
```

実験結果(XL, YL)の値をも含めて、区間途中のLagrange補間した値は次のように得られる。

```
   XL
_3 _1 0 2 5
   YL
_18 _8 10 50 26
   (XL;YL) lagr _3 _1 _0.5 0 0.5 1 1.5 2
_18 _8 0.314236 10 20.4781 31.1111 41.2031 50
```

３．Spline補間とそのプログラム

Spline補間では、まず準備として各測定点で区切って区間に分ける。そして、それぞれの区間内を３次の多項式（ふつうは３次のSpline）で別々に補間する。これらの区間ごとの補間の関数をつなぎ合わせたものがSpline補間である。これはかつての製図用の自在定規をまねたものと言われている。

３次多項式を一般的に表す。（ここでは昇べきであることに注意）

$f(x) = a + bx + cx^2 + dx^3$

これからそれぞれの区間で、関数値、１次微分、２次微分は次のようになる。

[-3, -1] では
$f_1(x) = a_1 + b_1x + c_1x^2 + d_1x^3$
$f_1'(x) = b_1 + 2c_1x + 3d_1x^2$
$f_1''(x) = 2c_1 + 6d_1x$

[-1, 0] では
$f_2(x) = a_2 + b_2x + c_2x^2 + d_2x^3$
$f_2'(x) = b_2 + 2c_2x + 3d_2x^2$
$f_2''(x) = 2c_2 + 6d_2x$

[0, 2] では
$f_3(x) = a_3 + b_3x + c_3x^2 + d_3x^3$
$f_3'(x) = b_3 + 2c_3x + 3d_3x^2$
$f_3''(x) = 2c_3 + 6d_3x$

[2, 5] では
$f_4(x) = a_4 + b_4x + c_4x^2 + d_4x^3$
$f_4'(x) = b_4 + 2c_4x + 3d_4x^2$
$f_4''(x) = 2c_4 + 6d_4x$

ここでSplineの条件とは、次のように定める。

- 測定点で、それぞれの関数値は測定値に等しい。 $f_1(-3) = -18, \cdots$
- 隣同士の１次微係数は等しい。 $f_1'(-1) = f_2'(-1), \cdots$
- 隣同士の２次微係数は等しい。 $f_2''(-1) = f_2''(-1), \cdots$
- 末端での２次微係数は０とする。 $f_2''(-3) = 0,\ f_2''(5) = 0$

これらの式に区間の値を入れると、未知数を$a_1, \cdots d_4$とする連立方程式が得られる。

$$a_1 + b_1(-3) + c_1(-3)^2 + d_1(-3)^3 = -18$$
$$a_1 + b_1(-1) + c_1(-1)^2 + d_1(-1)^3 = -8$$
$$a_2 + b_2(-1) + c_2(-1)^2 + d_2(-1)^3 = -8$$
$$a_2 + b_2\,0 + c_2\,0^2 + d_2\,0^3 = 10$$
$$a_3 + b_3\,0 + c_3\,0^2 + d_3\,0^3 = 10$$
$$a_3 + b_3\,2 + c_3\,2^2 + d_3\,2^3 = 50$$
$$a_4 + b_4\,2 + c_4\,2^2 + d_4\,2^3 = 50$$
$$a_4 + b_4\,5 + c_4\,5^2 + d_4\,5^3 = 26$$
$$b_1 + 2c_1(-1) + 3d_1(-1)^2 - (b_2 + 2c_2x(-1) + 3d_2x(-1)^2) = 0$$
$$b_2 + 2c_2\,0 + 3d_2\,0^2 - (b_3 + 2c_3\,0 + 3d_3\,0^2) = 0$$
$$b_3 + 2c_3\,2 + 3d_3\,2^2 - (b_4 + 2c_4\,2 + 3d_4\,2^2) = 0$$
$$2c_1 + 6d_1(-1) - (2c_2 + 6d_2(-1)) = 0$$
$$2c_2 + 6d_2\,0 - (c_3 + 6d_3\,0) = 0$$
$$2c_3 + 6d_3\,2 - (2c_4 + 6d_4\,2) = 0$$
$$2c_1 + 6d_1(-3) = 0$$
$$2c_4 + 6d_4\,5 = 0$$

この連立方程式を解くことは、Lagrange補間の場合と同様、Jにとっては、ごく容易なことに過ぎない。直ちに、Jのプログラミングに入る。

測定値データを次のようにする。

```
XS =: _3  _1  0  2  5
YS =: _18 _8 10 50 26
```

これら測定値から、行列の要素としてまとめるのに、それぞれプログラムを作った。
まず、関数値については、J の動詞 f0 と f00 により行う。

```
f0 =: 3 : 0
X =. y.
N =. <: #X
}. }: 2 # <"(1) y. ^/ (i. <: #X)
)
```

```
   f0 XS
+----------+--------+--------+-------+-------+-------+-------+----------+
|1 _3 9 _27|1 _1 1 _1|1 _1 1 _1|1 0 0 0|1 0 0 0|1 2 4 8|1 2 4 8|1 5 25 125|
+----------+--------+--------+-------+-------+-------+-------+----------+
```

```
f00 =: 3 : 0
PPA =: (8, 4)$(<4#0)
PI0 =: f0 XS
NSA =. +: NS - 1
i =. 0
j =. 0
while. i < NSA
  do.
  j =. <. -: i
  wr i, j
NB.  wr (i{PI0)
  PPA =: (i{PI0) (<i, j) } PPA
  i =. i + 1
  end.
)
```

```
   PPA
+----------+--------+-------+----------+
|1 _3 9 _27|0 0 0 0 |0 0 0 0|0 0 0 0   |
+----------+--------+-------+----------+
|1 _1 1 _1 |0 0 0 0 |0 0 0 0|0 0 0 0   |
+----------+--------+-------+----------+
|0 0 0 0   |1 _1 1 _1|0 0 0 0|0 0 0 0   |
+----------+--------+-------+----------+
|0 0 0 0   |1 0 0 0 |0 0 0 0|0 0 0 0   |
+----------+--------+-------+----------+
|0 0 0 0   |0 0 0 0 |1 0 0 0|0 0 0 0   |
+----------+--------+-------+----------+
|0 0 0 0   |0 0 0 0 |1 2 4 8|0 0 0 0   |
+----------+--------+-------+----------+
|0 0 0 0   |0 0 0 0 |0 0 0 0|1 2 4 8   |
+----------+--------+-------+----------+
|0 0 0 0   |0 0 0 0 |0 0 0 0|1 5 25 125|
+----------+--------+-------+----------+
```

次に、 １次微係数については、動詞 f1 と f11 により行う。

```
PPB =: (3, 4)$(<4#0)

K1 =: }: 2#0 1 2 3 4
K2 =: }. 2#0 1 2 3 4
KK =: K1,. K2

f1 =: 3 : 0
X_1 =. }. }: y.
PA_1 =. 0,"(1) (1, 2, 3) *"(1) y. ^/ (i.  #X_1)
PB_1 =. 2 # PA_1
M_1 =. $PB_1
PC_1 =. ,|: (-: {. M_1)#"(0) (1, _1)
}. }: <"(1) ((-: {. M_1), (+: {: M_1)) $ , PC_1 *"(0 1) PB_1
)

f11 =: 3 : 0
P11 =: <"(1)(6 4)$,> f1 XS
NSB =: +: NS - 2
i =. 0
while. i < NSB
  do.
NB.      wr i { KK
  PPB =: (i{P11) (<i{KK) } PPB
  i =. i + 1
  end.
)
   PPB
+--------+---------+--------+-----------+
|0 1 _2 3|0 _1 2 _3|0 0 0 0 |0 0 0 0    |
+--------+---------+--------+-----------+
|0 0 0 0 |0 1 0 0  |0 _1 0 0|0 0 0 0    |
+--------+---------+--------+-----------+
|0 0 0 0 |0 0 0 0  |0 1 4 12|0 _1 _4 _12|
+--------+---------+--------+-----------+
```

同様にして、２次微係数については、動詞 f2 と f22 により行う。

```
f2 =: 3 : 0
X_2 =: }. }: y.
PA_2 =. (0, 0),"(1) (2, 6) *"(1) y. ^/ (i. <: #X_2)
PB_2 =. 2 # PA_2
M_2 =. $PB_2
PC_1 =. ,|: (-: {. M_2)#"(0) (1, _1)
}. }: <"(1) ((-: {. M_2), (+: {: M_2)) $ , PC_1 *"(0 1) PB_2
)
PPC =: (3, 4)$(<4#0)
f22 =: 3 : 0
```

```
P22 =: <"(1)(6 4)$,> f2 XS
NSB =: +: NS - 2
i =. 0
while. i < NSB
  do.
     wr i { KK
  PPC =: (i{P22) (<i{KK) } PPC
  i =. i + 1
  end.
)
   PPC
+--------+--------+--------+----------+
|0 0 2 _6|0 0 _2 6|0 0 0 0 |0 0 0 0    |
+--------+--------+--------+----------+
|0 0 0 0 |0 0 2 0 |0 0 _2 0|0 0 0 0    |
+--------+--------+--------+----------+
|0 0 0 0 |0 0 0 0 |0 0 2 12|0 0 _2 _12|
+--------+--------+--------+----------+
```

さらに、区間の両端部分については、動詞 f3 と f33 とした。

```
PPD =: (2, 4)$(<4#0)
f3 =: 3 : 0
X_3A =. {. y.
X_3B =. {: y.
PA_3 =. (0, 0),"(1) (2, 6) *"(1) (1, X_3A)
PB_3 =. (0, 0),"(1) (2, 6) *"(1) (1, X_3B)
PA_3;PB_3
)

f33 =: 3 : 0
P33 =: f3 XS
PPD =. (0{P33) (<0, 0) } PPD
PPD =: (1{P33) (<1, 3) } PPD
)

   PPD
+---------+-------+-------+--------+
|0 0 2 _18|0 0 0 0|0 0 0 0|0 0 0 0 |
+---------+-------+-------+--------+
|0 0 0 0  |0 0 0 0|0 0 0 0|0 0 2 30|
+---------+-------+-------+--------+
```

これらを連結して、連立方程式の係数行列はつぎのように得られる。

```
   PMA =: 16 16$, > PPA, PPB, PPC, PPD
   PMA
1 _3  9 _27 0  0  0  0 0  0  0  0 0  0  0   0
1 _1  1  _1 0  0  0  0 0  0  0  0 0  0  0   0
0  0  0   0 1 _1  1 _1 0  0  0  0 0  0  0   0
0  0  0   0 1  0  0  0 0  0  0  0 0  0  0   0
0  0  0   0 0  0  0  0 1  0  0  0 0  0  0   0
0  0  0   0 0  0  0  0 1  2  4  8 0  0  0   0
0  0  0   0 0  0  0  0 0  0  0  0 1  2  4   8
0  0  0   0 0  0  0  0 0  0  0  0 1  5 25 125
```

```
0  1 _2   3 0 _1  2 _3 0  0  0  0 0  0  0   0
0  0  0   0 0  1  0  0 0 _1  0  0 0  0  0   0
0  0  0   0 0  0  0  0 0  1  4 12 0 _1 _4 _12
0  0  2  _6 0  0 _2  6 0  0  0  0 0  0  0   0
0  0  0   0 0  0  2  0 0  0 _2  0 0  0  0   0
0  0  0   0 0  0  0  0 0  0  2 12 0  0 _2 _12
0  0  2 _18 0  0  0  0 0  0  0  0 0  0  0   0
0  0  0   0 0  0  0  0 0  0  0  0 0  2  30
```

また、連立方程式の右辺 YDA は、測定値 YS から次のようになる。

```
   YDA =: }. }: 2#YS
   YDA =: YDA, 8#0
   YDA
_18 _8 _8 10 10 50 50 26 0 0 0 0 0 0 0 0
```

連立方程式の解は J のプリミティブ %. により一発で得られる。

```
   YDA %. PMA
12 28 9 1 10 22 3 _1 10 22 3 _2 _14 58 _15 1
```

すなわち、$a_1 = 12,\ b_1 = 28,\ c_1 = 9,\ d_1 = 1,$

$a_2 = 10,\ b_2 = 22,\ c_2 = 3,\ d_2 = -1,$

$a_3 = 10,\ b_3 = 22,\ c_3 = 3,\ d_3 = -2,$

$a_4 = 14,\ b_4 = 58,\ c_4 = -15,\ d_4 = 1$

また、区間外 [-∞, -3]では Spline は次のような直線となる。

傾き $f'_1(-3) = b_1 + 2c_1(-3) + 3d_1(-3)^2 = 1$ と 点(_3, _18)を通ることから

$y = 1(x - \_3) + \_18$ つまり $y = x - 15$

同様に、区間外 [5, ∞]では Spline は次のような直線となる。

傾き $f'_4(5) = b_4 + 2c_4(5) + 3d_4(5)^2 = -17$ と 点(5, 26)を通ることから

$y = -17(x - 5) + 26$ つまり $y = -17x + 111$

最後の結果である Spline 補間の式は J のプログラムとして spl として定義される。ここで、補助関数 int は求めたい数値に対して、左引数 XS の区間を参照して、何番目の Spline 補間式を使うかを選択させるものである。

```
int =: 3 : 0
:
(* x. - y.) i. 1
)

spl =: 3 : 0"(0)
FN =. XS int y.
select. FN
  case. 0 do. func =: _15 + y.
  case. 1 do. func =: 12  + (28*y.) + (9*y.^2) + (y.^3)
  case. 2 do. func =: 10  + (22*y.) + (3*y.^2) + (- y.^3)
  case. 3 do. func =: 10  + (22*y.) + (3*y.^2) + (-2*y.^3)
  case. 4 do. func =: _14 + (58*y.) + (-15*y.^2) + (y.^3)
  case. 5 do. func =: 111 + (-17*y.)
end.
)
```

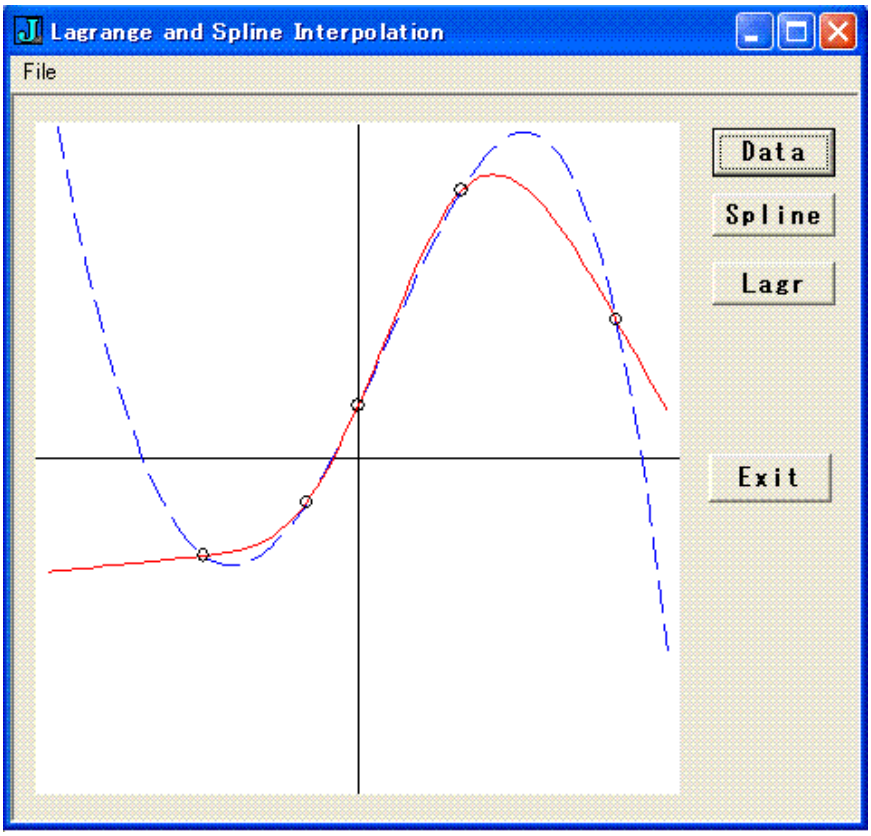
実行は次のように行われる。

```
   spl _4 _3 _2 _1 0 1 2 3 4 5 6
_19 _18 _16 _8 10 33 50 52 42 26 9
```

４．Lagrange 補間と Spline 補間のグラフ表示

実際のグラフ表示は次のようになる。
ここでは、両者を比較するため、gl2 によるウィンドウ・グラフィックスを用いた。簡便な plot ルーチンでもちろん可能である。

Lagrange 補間(点線)が両端で波うってしまうのに対して、Spline 補間(実線)が極めて滑らかな曲線で各点を結んでいることが見られる。それぞれボタン[Data], [Lagr], [Spline]を押すことで、別々に表示、比較できるようにした。



５．Brian Bradie の例題への適用

Brian Bradie の教科書[2]の中に Lagrange p.341-346 および Spline p.393-397 の記載があった。しかし、説明は必ずしもわかり易くはない。

前節までの原理的な説明とプログラムを Brian Bradie のデータにも適用できるように、先のプログラムを汎用的に改良した。プログラム・リストは稿末に載せた。

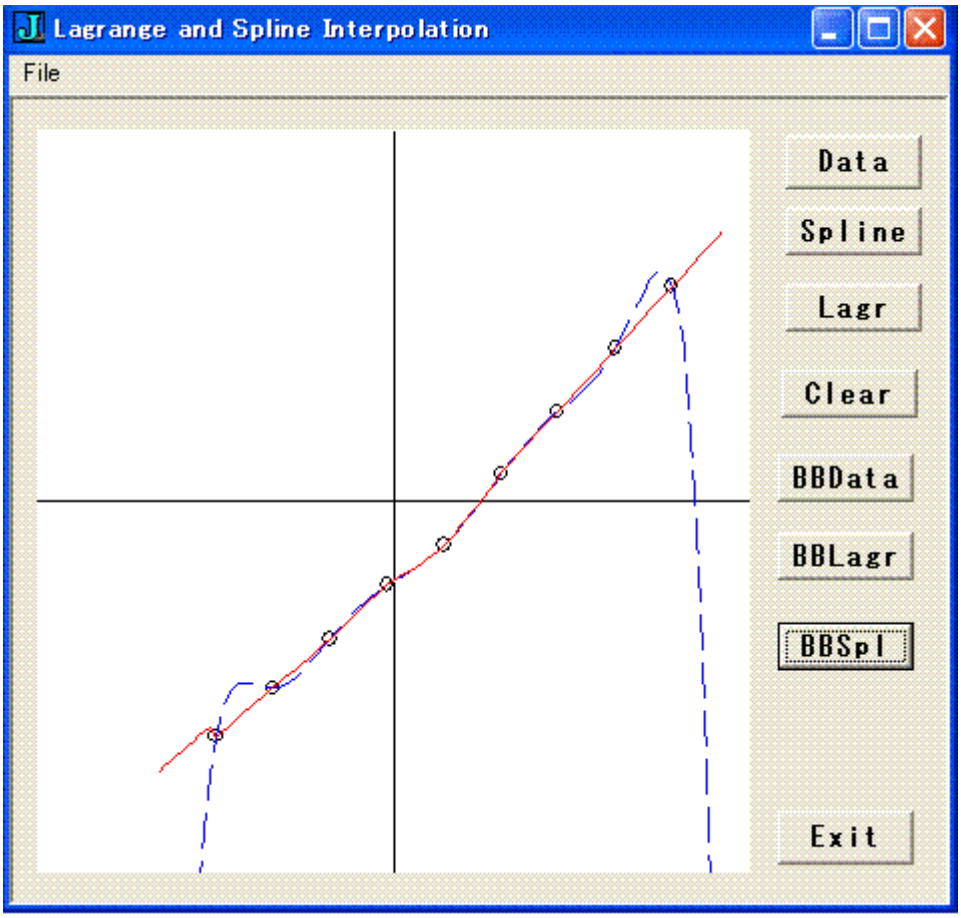
X	300	400	500	600	700	800	900	1000	1100
Y	0.024	0.035	0.046	0.058	0.067	0.083	0.097	0.111	0.125

なお、 Brian Bradie のオリジナルな値ではなく、以下のように調整した値を用いた。

X	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1	1.1
Y	0.24	0.35	0.46	0.58	0.67	0.83	0.97	1.11	1.25

[2] Brian Bradie, "A Friendly Introduction of Numerical Analysys"
Pearson Education International, Prentice Hall (2006).

結果のグラフ表示は次のようになる。Bradie の書にあるグラフと比較されたい。やはり、Langrange 補間では、区間の両端部分ではうまくいかない。Spline では滑らかに補間されている。



６．おわりに－Ｊがあれば Calculus（数式計算学）は要らない！

マクロな（目に見える、原子、分子レベルではない）自然現象はすべてアナログな幾何学図形として起る。これを人間が処理するためにディジタルという道具を考え出し、さらに一般的に行うため式による計算学を作った。これが Calculus であり、日本では微分積分学と言っているがその一部でしかない。

われわれにはＪという強力な道具がある。すなわち、Ｊには式の計算を行わずして、これをやってくれる数々のプリミティブがある。たとえば、

%.（連立方程式の根）、p:（高次方程式の根）、
q:（素因数分解）、C.（群論の置換群操作）

などである。さらに、結果を図で見せてくれる強力なグラフィックスがある。

すべからず、われわれは従来の数式演算の「くびき」から脱却すべきである。現在、平方根を手計算で行う人などいないのと同様、式による長々とした計算(derivation)をありがたがる数学教育は過去のものでしかない。原理に基づく、アルゴリズムこそポイントである。そして、何々の公式、誰々の方法などと、いたずらに処理をブラックボックス化させないことが大切である、と私は思う。

```
NB. Brian Bradie / Friendly Introduction of Numerical Analysis p.396 ====
XBB_Orig =: 300 400 500 600 700 800 900 1000 1100
YBB_Orig =: 0.024 0.035 0.046 0.058 0.067 0.083 0.097 0.111 0.125

NB. Lagrange Interpolation ================================
XBL =: 100 %~ XBB_Orig
YBL =: 100 *  YBB_Orig
NB.    (XBL;YBL) lagr 3 3.5 4 4.5 5
NB. 2.39999 3.60854 3.49998 3.85706 4.60003

NB. Spline Interpolation ==================================
XBS =: 1000 %~ XBB_Orig
YBS =: 10 * YBB_Orig

NB. Brian Bradie / Make Matrix
================================================
PB0 =: ff0 XBS
PB1 =: f1 ff XBS
PB2 =: f2 ff XBS
PB3 =: ff3 XBS

PBM =: PB0, PB1, PB2, PB3
PBA =: (32 32)$ , > PBM

YBX =: }. }: 2#YBS
YBX =: YBX, 16#0

NB. Brian Bradie / Calc. Spline coefficients
==================================
B_COEF0 =: YBX %. PBA
B_COEF1 =: 8 4$B_COEF0

NB. for [-∞, (0{XBS)] coefficient
B_AA =. +/ (}. 0{B_COEF1) * (1, 2, 3) * (0{XBS) ^ (i.3)
B_BA =. (0{YBS) - AA * (0{XBS)
B_COEF2 =: (B_BA, B_AA, 0, 0), B_COEF1

NB. for [(NS1{XBS), ∞] coefficient
NB1 =: (#XBS) - 1
B_AZ =. +/ (}. (NB1-1){B_COEF1) * (1, 2, 3) * (NB1{XBS) ^ (i.3)
B_BZ =. (NB1{YBS) - B_AZ * (NB1{XBS)
B_COEF3 =: B_COEF2, (B_BZ, B_AZ, 0, 0 )
B_COEF =: B_COEF3

NB. Brian Bradie / Calc. Spline
================================================
NB.   bb_spli 0.2 0.25 0.3 0.35 0.4 0.45 0.5
NB. 0.163367 0.219209 0.24 0.295631 0.35 0.403106 0.46
NB. XBS, B_COEF are global, using int
bb_spli =: 3 : 0"(0)
XC =. y.^(i.4)
FN =. XBS int y.
(FN { B_COEF) (+ / . *) XC
)
```

Jのプログラム・リスティング

```
NB. Langrange Interpolation ====================
wr =: 1!:2&2

NB. 戸田英雄、小野令美「入門数値計算」p.35 オーム社

XL =: 1 2 4 5
YL =: 3 2 12 35
NL =: |. i. #XL
AL =: |: (|. i. #XL) (^~/) XL
NB. AL =: 4 4$1 1 1 1 8 4 2 1 64 16 4 1 125 25 5 1
FL =: YL %. AL
NB. 1 _5 7 0

NB. lag 2 => 2
NB. lag 2.5 => 1.875
NB. lag 3 => 3
NB. lag 4 => 12

lag =: 3 : 0"(0)
XX =. NL ^~ y.
+/ FL * XX
)


NB.    (XL;YL) lagr 2.5 => 1.875
NB.     (XL;YL) lagr 1 1.5 2 2.5 3 3.5 4 4.5 5
NB. 3 2.625 2 1.875 3 6.125 12 21.375 35
lagr =: 3 : 0"(1 0)
:
'X Y' =. x.
N =. |. i. #X
A =. |: (|. i. #X) (^~/) X
F =. Y %. A
XX =. N ^~ y.
+/ F * XX
)




NB. Spline Interpolation ======================
NB. 戸田英雄、小野令美「入門数値計算」p.40 オーム社

NB. test ==============================================
P_1 =: (1, _3, 9, _27), 12#0
```

```
P_2 =: (1, _1, 1, _1),  12#0
P_3 =: (4#0), (1, _1, 1, _1), 8#0
P_4 =: (4#0), 1, (11#0)

P_5 =: (8#0), 1, (7#0)
P_6 =: (8#0), (1 2 4 8), (4#0)
P_7 =: (12#0), (1 2 4 8)
P_8 =: (12#0), (1 5 25 125)

P_9 =: (0 1 _2 3 0 _1 2 _3), (8#0)
P_10 =: (4#0), (0 1 0 0), (0 _1 0 0), (4#0)
P_11 =: (8#0), (0 1 4 12), (0 _1 _4 _12)

P_12 =: (0 0 2 _6), (0 0 _2 6), (8#0)
P_13 =: (4#0), (0 0 2 0), (0 0 _2 0), (4#0)
P_14 =: (8#0), (0 0 2 12), (0 0 _2 _12)

P_15 =: (0 0 2 _18), (12#0)
P_16 =: (12#0), (0 0 2 30)

P_P =: P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8, P_9, P_10, P_11, P_12, P_13,
P_14, P_15,: P_16

X_X =: (_18 _8 _8 10 10 50 50 26), (8#0)

NB.    X_X %. P_P
NB. 12 28 9 1 10 22 3 _1 10 22 3 _2 _14 58 _15 1

NB. Spline Program
================================================================

XS =: _3  _1  0  2  5
YS =: _18 _8 10 50 26

NS =: #XS

NB.    f0 XS
NB. +----------+---------+---------+-------+-------+-------+-------
+----------+
NB. |1 _3 9 _27|1 _1 1 _1|1 _1 1 _1|1 0 0 0|1 0 0 0|1 2 4 8|1 2 4 8|1 5 25
125|
NB. +----------+---------+---------+-------+-------+-------+-------
+----------+
f0 =: 3 : 0
X =. y.
```

```
N =. <: #X
}. }: 2 # <"(1) y. ^/ (i. <: #X)
)

NB.    f1 XS
NB. +----------------+---------------+-------------------+
NB. |0 1 _2 3 0 _1 2 _3|0 1 0 0 0 _1 0 0|0 1 4 12 0 _1 _4 _12|
NB. +----------------+---------------+-------------------+
f1 =: 3 : 0
X_1 =. }. }: y.
PA_1 =. 0,"(1) (1, 2, 3) *"(1) y. ^/ (i.  #X_1)
PB_1 =. 2 # PA_1
M_1 =. $PB_1
PC_1 =. ,|: (-: {. M_1)#"(0) (1, _1)
}. }: <"(1) ((-: {. M_1), (+: {: M_1)) $ , PC_1 *"(0 1) PB_1
)

NB.   f2 XS
NB. +----------------+---------------+------------------+
NB. |0 0 2 _6 0 0 _2 6|0 0 2 0 0 0 _2 0|0 0 2 12 0 0 _2 _12|
NB. +----------------+---------------+------------------+
f2 =: 3 : 0
X_2 =: }. }: y.
PA_2 =. (0, 0),"(1) (2, 6) *"(1) y. ^/ (i. <: #X_2)
PB_2 =. 2 # PA_2
M_2 =. $PB_2
PC_1 =. ,|: (-: {. M_2)#"(0) (1, _1)
}. }: <"(1) ((-: {. M_2), (+: {: M_2)) $ , PC_1 *"(0 1) PB_2
)

NB.    f3 XS
NB. +---------+--------+
NB. |0 0 2 _18|0 0 2 30|
NB. +---------+--------+
f3 =: 3 : 0
X_3A =. {. y.
X_3B =. {: y.
PA_3 =. (0, 0),"(1) (2, 6) *"(1) (1, X_3A)
PB_3 =. (0, 0),"(1) (2, 6) *"(1) (1, X_3B)
PA_3;PB_3
)

NB. Make Matrix ================================
PP =: ((4 * (NS - 1)), 4)$(<4#0)
PPA =: (8, 4)$(<4#0)
```

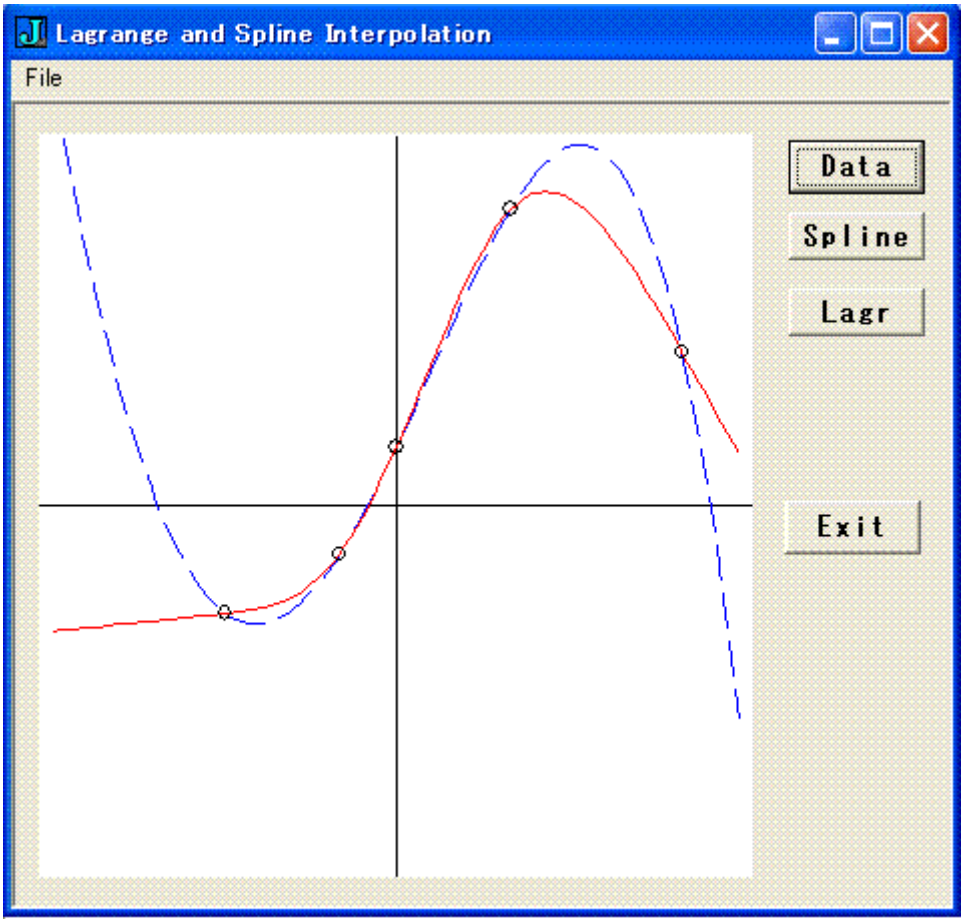
```
f00 =: 3 : 0
PI0 =: f0 XS
NSA =. +: NS - 1
i =. 0
j =. 0
while. i < NSA
  do.
  j =. <. -: i
  wr i, j
NB.  wr (i{PI0)
  PPA =: (i{PI0) (<i, j) } PPA
  i =. i + 1
  end.
)

PPB =: (3, 4)$(<4#0)

K1 =: }: 2#0 1 2 3 4
K2 =: }. 2#0 1 2 3 4
KK =: K1,. K2

f11 =: 3 : 0
P11 =: <"(1)(6 4)$,> f1 XS
NSB =: +: NS - 2
i =. 0
while. i < NSB
  do.
     wr i { KK
  PPB =: (i{P11) (<i{KK) } PPB
  i =. i + 1
  end.
)

PPC =: (3, 4)$(<4#0)

f22 =: 3 : 0
P22 =: <"(1)(6 4)$,> f2 XS
NSB =: +: NS - 2
i =. 0
while. i < NSB
  do.
     wr i { KK
  PPC =: (i{P22) (<i{KK) } PPC
  i =. i + 1
```

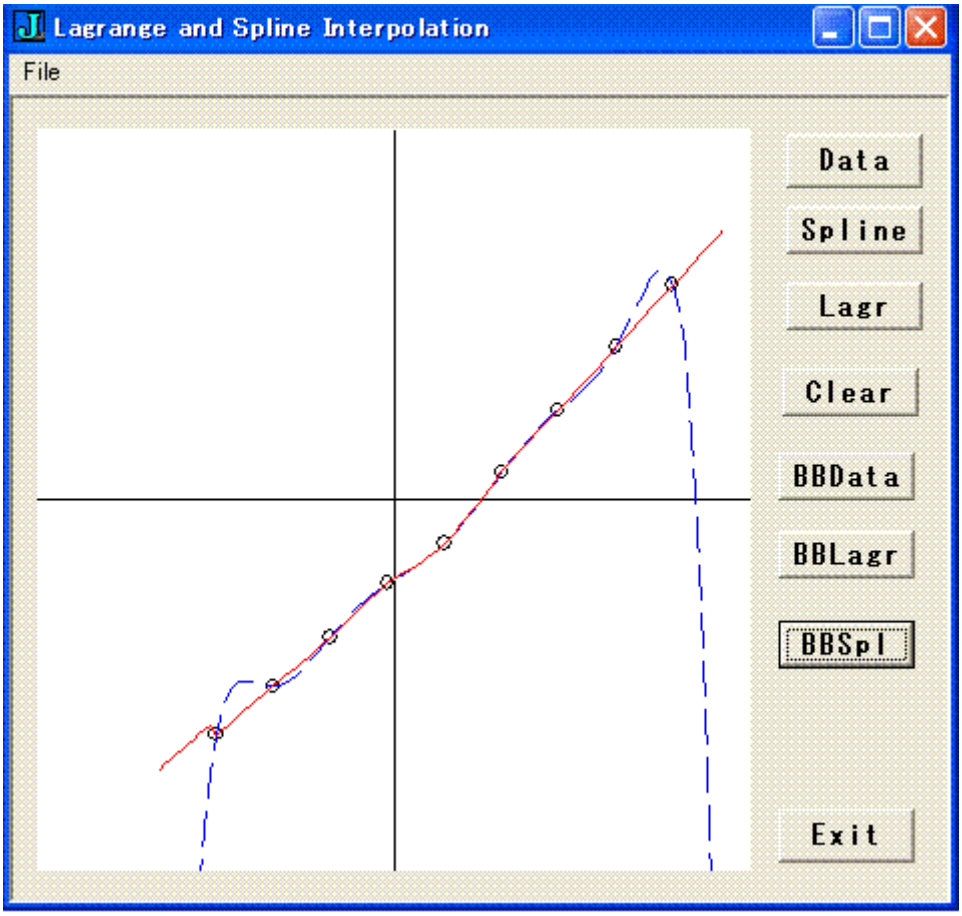
```
  end.
)

PPD =: (2, 4)$(<4#0)

f33 =: 3 : 0
P33 =: f3 XS
PPD =. (0{P33) (<0, 0) } PPD
PPD =: (1{P33) (<1, 3) } PPD
)

int =: 3 : 0
:
(* x. - y.) i. 1
)

spl =: 3 : 0"(0)
FN =. XS int y.
select. FN
  case. 0 do. func =: _15 + y.
  case. 1 do. func =: 12 + (28*y.) + (9*y.^2) + (y.^3)
  case. 2 do. func =: 10 + (22*y.) + (3*y.^2) + (- y.^3)
  case. 3 do. func =: 10 + (22*y.) + (3*y.^2) + (-2*y.^3)
  case. 4 do. func =: _14 + (58*y.) + (-15*y.^2) + (y.^3)
  case. 5 do. func =: 111 + (-17*y.)
end.
)

NB. Spline Revised Version enable for Toda and Bradie ==========
NB. PP =: ff0 XS
ff0 =: 3 : 0
XX =. y.
NN =: #XX
PP0 =: ((+: NN - 1), (NN - 1))$(<4#0)
PI0 =: f0 XX
NS0 =: +: NN - 1
i =. 0
j =. 0
while. i < NS0
  do.
  j =. <. -: i
NB.  wr i, j
NB.  wr (i{PI0)
  PP0 =. (i{PI0) (<i, j) } PP0
  i =. i + 1
```

```
  end.
PP0
)

NB. PP1 =. f1 ff XB
NB. PP2 =. f2 ff XB
NB. ff is defined adverb, f1, f2 as left argument verb
ff =: 1 : 0
XX =. y.
NN =. #XX
PP1 =: ((NN - 2), (NN - 1))$(<4#0)
PI1A =: <"(1) ((+: NN -2), (NN - 1))$,> u. XX
PI1 =. <"(1) ((+: {. $ PI1A), 4)$,>PI1A
NS1 =. +: NN -2
K1 =. }: 2#(i.NN)
K2 =. }. 2#(i.NN)
KK =. K1,. K2
i =. 0
while. i < NS1
  do.
NB.  wr i { KK
  PP1 =. ( {. i{PI1) (< i{KK) } PP1
  i =. i + 1
  end.
PP1
)


NB. Graphics ========================================

require 'gl2'

SPLINE=: 0 : 0
pc spline;pn "Lagrange and Spline Interpolation";
menupop "File";
menu new "&New" "" "" "";
menu open "&Open" "" "" "";
menusep ;
menu exit "&Exit" "" "" "";
menupopz;
xywh 275 7 34 12;cc ok button;cn "Data";
xywh 274 101 34 12;cc cancel button;cn "Exit";
xywh 6 7 259 224;cc Interp isigraph;
xywh 274 24 34 11;cc sp button;cn "Spline";
xywh 274 43 34 11;cc lagran button;cn "Lagr";
```

```
pas 6 6;pcenter;
rem form end;
)

run =: spline_run
spline_run=: 3 : 0
wd SPLINE
NB. initialize form here
gllines 0 500 1000 500
gllines 500 0 500 1000
glshow ''
wd 'pshow;'
)

spline_close=: 3 : 0
wd'pclose'
)

spline_cancel_button=: 3 : 0
spline_close''
)

adj0 =: 3 : '500 + 100 * y.'
DXY =: 2 3$_1 2 3 10 25 30
adjxy =: 3 : ', |: 500 + (80, 8) * y.'

gcircle =: 3 : 0
:
R =. x.         NB. radius
'X0 Y0' =. y.   NB. center
X1 =. X0-R
Y1 =. Y0-R
R2 =. 2*R
NB.    x   y   w   h   stx sty edx edy
glarc X1, Y1, R2, R2, X1, Y0, X1, Y0
glshow ''
)

spline_ok_button=: 3 : 0
glrgb 0 0 0
glpen 1 0
NB. gllines adjxy XS,:YS
NB. 10 gcircle"(0 1) > 300 400;600 800;700 600
NB. 10 gcircle"(0 1) 500 + (80, 8) * "(1) |: XS,:YS
10 gcircle"(0 1) 500 + (80, 8) * "(1) XS,.YS
```

```
glshow ''
)

XD =: (i: 60) % 10   NB. generate plot_data

NB. Spline
spline_sp_button=: 3 : 0
NB. gllines adjxy (] ,: spl) _1 _0.9 _0.8
glrgb 255 0 0
glpen 1 0
gllines adjxy (] ,: spl) XD    NB. draw spline interpolation
glshow ''
)

XYL =: (XS;YS) lagr XD
NB. Lagrange
spline_lagran_button=: 3 : 0
glrgb 0 0 255
glpen 1 0
gllines adjxy (XD,: XYL)    NB. draw Lagrangean interpolation
glshow ''
)
```




Lagrange and Spline Interpolation
File
Data
Spline
Lagr
Clear
BBData
BBLagr
BBSpl
Exit