数理解析研究所講究録
第613巻 1987年 188-203

ベクトル計算機に適したB－スプラインの計算法

明石高専　吉本富士市（Fujiichi Yoshimoto）
京大・工　津田孝夫（Takao Tsuda）

1．はじめに

スプライン関数は，補間，データ平滑化，積分，画像処理，ＣＡＧＤなど多くの分野で用いられている重要な近似関数である．スプライン関数を用いるとき，今日では，ほとんどの場合Ｂ－スプラインを基底関数としている．したがって，Ｂ－スプラインの高速な計算法を確立することは，大きな意義があると思われる．

ここでは，ベクトル計算機に適したＢ－スプラインの計算法を提案し，それを用いた補間，データ平滑化の計算例を示すことにしたい．

2．Ｂ－スプラインの計算法

2．1 従来の計算法[1)]

いま，x 軸上の実数列を $\xi_{j-m} \leq \cdots \leq \xi_{j-1} \leq \xi_j$ とし，切断べき関数を

$$M_m(x;y) = (y-x)_+^{m-1} = \begin{cases} (y-x)^{m-1} & y > x \\ 0 & y \leq x \end{cases} \qquad (1)$$

と表すとき，m 階（$m-1$ 次）のＢ－スプラインは次の式で定義される．

$$M_m(x) = M_m(x;\ \xi_{j-m}, \cdots, \xi_{j-1}, \xi_j). \qquad (2)$$

ここで，（2）の右辺は切断べき関数の $y = \xi_{j-m}, \cdots, \xi_{j-1}, \xi_j$ に関するm階の差分商を意味する．すなわち，

$$M_m(x;\ \xi_{j-r}, \cdots, \xi_{j-1}, \xi_j) = \frac{M_m(x;\ \xi_{j-r+1}, \cdots, \xi_j) - M_m(x;\ \xi_{j-r}, \cdots, \xi_{j-1})}{\xi_j - \xi_{j-r}}$$

$$(r = 1, 2, \cdots, m) \qquad (3)$$

である．実数列 $\xi_{j-m}, \cdots, \xi_{j-1}, \xi_j$ は節点と呼ばれている．

ところが，（3）をそのまま用いてＢ－スプラインの値を計算すると，節点の近くでは桁落ちのため高精度な計算ができない．そこで，通常は次のｄｅＢｏｏｒ－Ｃｏｘのアルゴリズムを用いて計算している．いま，

$$M_{1j}(x) = \begin{cases} (\xi_j - \xi_{j-1})^{-1} & (\xi_{j-1} \leqq x < \xi_j) \\ 0 & (\text{その他}) \end{cases} \qquad (4)$$

とするとき，このアルゴリズムは，

$$M_{rj}(x) = \frac{(x - \xi_{j-r}) M_{r-1,j-1}(x) + (\xi_j - x) M_{r-1,j}(x)}{\xi_j - \xi_{j-r}}$$

$$(r = 2, 3, \cdots, m) \qquad (5)$$

と書くことができる．

さて，Ｂ－スプラインは次の性質をもっている．

$$M_{mj}(x) = \begin{cases} > 0 & (\xi_{j-m} < x < \xi_j) \\ = 0 & (\text{その他}), \end{cases} \qquad (6)$$

$$\int_{-\infty}^{\infty} M_{mj}(x)\, dx = \frac{1}{m}. \qquad (7)$$

このことから，節点の間隔が小さいとき $M_{mj}(x)$ は大きなピーク値をもち，反対に節点の間隔が大きいとき小さなピーク値をもつことが分かる．このため（5）の線形結合で近似関数を構成すると，補間，あてはめなどの問題で，解くべき連立一次方程式が条件の良いものにならないことが多い．そこで，次のような正規化を行っている．

$$N_{mj}(x) = (\xi_j - \xi_{j-m}) M_{mj}(x). \qquad (8)$$

さて，x 軸上の区間［a，b］で，（8）を基底関数として用いた近似関数を作ることにしよう．いま，区間［a，b］の内部の節点を $\xi_1, \xi_2, \cdots, \xi_{k-m}$（$\xi_1 \leqq \xi_2 \leqq \cdots \leqq \xi_{k-m}$）とし，両端 a，b で m 個ずつの付加節点を置く．すなわち，

$$\left.\begin{aligned} a &= \xi_{1-m} = \cdots = \xi_{-1} = \xi_0 \\ b &= \xi_{k-m+1} = \cdots = \xi_{k-1} = \xi_k \end{aligned}\right\} \qquad (9)$$

とすると，近似関数は，

$$S(x) = \sum_{j=1}^{h} c_j N_{m,j}(x) \qquad (10)$$

となる．式（６），（８），（１０）から，区間［a，b］内の任意の点xに対してm個のＢ－スプラインの値を計算する必要があることになる．

これらの値は，（４），（５）から次のようにして計算できる．いま，$\xi_{j-1} \leqq x < \xi_j$とする．まず，下記の配列の要素を計算する．

$$\begin{array}{ccccc}
 & & & & M_{m-1,j}(x) \\
 & & M_{3,j}(x) & & \\
 & M_{2,j}(x) & & & M_{m-1,j+1}(x) \\
M_{1,j}(x) & & M_{3,j+1}(x) & & \vdots \\
 & M_{2,j+1}(x) & & & \\
 & & M_{3,j+2}(x) & & \\
 & & & & M_{m-1,j+m-2}(x)
\end{array} \qquad (11)$$

$M_{1,j}(x)$から計算を始め，左から右へと順次計算する．このとき，各要素は，その要素の属する列のすぐ左側の二つの要素（一つはすぐ上の要素，もう一つはすぐ下の要素）を用いて計算できる．たとえば，$M_{3,j+1}(x)$は$M_{2,j}(x)$と$M_{2,j+1}(x)$を用いて計算できる．また，（６）の性質から，この配列に表れない要素はすべて零であるから，たとえば$M_{3,j+2}(x)$は$M_{2,j+1}(x)$のみから計算できる．

次に，（５），（８）より

$$N_{m,j}(x) = (x - \xi_{j-m}) M_{m-1,j-1}(x) + (\xi_j - x) M_{m-1,j}(x) \qquad (12)$$

であるから，（１１）の最も右側の値を（１２）へ代入すると，$\xi_{j-1} \leqq x < \xi_j$となる$x$に対して，必要な$m$個のＢ－スプラインの値$N_{m,i}(x)$（$i = j, j+1, \cdots, j+m-1$）を求めることができる．

さて，以上の計算法に基ずいてＢ－スプラインを計算するプログラムを作成すると，たとえば図１のようになる．このプログラムで，変数ＲＭ，ＲＮはそれぞれ（１１）のM，（１２）のNに対応している．また，変数ＩＯＲはＢ－スプラインの階数mを意味している．さらに，変数Ｘ，ＧＺＡＩはそれぞれＢ－スプラインの値を計算したい点xおよび節点ξである．

```
      SUBROUTINE BSPL (X,NKNOT,GZAI,IOR,IGZ,RN)
C     COMPUTATION OF THE VALUES OF NORMALIZED B-SPLINES
      DIMENSION GZAI(100),RM(9,9),RN(10)
      REAL*8 X,GZAI,RM,RN
      DO 10 I=IOR+1,NKNOT-IOR
      IF(X .LT. GZAI(I)) THEN
        IGZ=I
        GO TO 20
      END IF
   10 CONTINUE
      IGZ=NKNOT-IOR+1
   20 CONTINUE
      RM(1,1)=1.0/(GZAI(IGZ)-GZAI(IGZ-1))
      DO 30 J=2,IOR-1
      RM(1,J)=(GZAI(IGZ)-X)*RM(1,J-1)/(GZAI(IGZ)-GZAI(IGZ-J))
      DO 40 I=2,J-1
      L=IGZ+I-1
      K=L-J
      RM(I,J)=((X-GZAI(K))*RM(I-1,J-1)+(GZAI(L)-X)*RM(I,J-1))/
     1        (GZAI(L)-GZAI(K))
   40 CONTINUE
      RM(J,J)=(X-GZAI(IGZ-1))*RM(J-1,J-1)/(GZAI(IGZ+J-1)-GZAI(IGZ-1))
   30 CONTINUE
      RN(1)=(GZAI(IGZ)-X)*RM(1,IOR-1)
      DO 50 I=2,IOR-1
      L=IGZ+I-1
      K=L-IOR
      RN(I)=(X-GZAI(K))*RM(I-1,IOR-1)+(GZAI(L)-X)*RM(I,IOR-1)
   50 CONTINUE
      RN(IOR)=(X-GZAI(IGZ-1))*RM(IOR-1,IOR-1)
      DO 60 I=1,IOR
      IF(RN(I) .LT. 1.0E-10) RN(I)=0.0
   60 CONTINUE
      RETURN
      END
```

図 1　Ｂ－スプラインの計算プログラム　（従来の計算法）

このプログラムをスカラ計算機　ＦＡＣＯＭ　Ｍ－３８２（以下Ｍ－３８２という）とベクトル計算機　ＦＡＣＯＭ　ＶＰ－２００（以下ＶＰ－２００という）にかけて，そのＣＰＵ時間の比を求めると，図２のようになる．この図から分かるように，図１のプログラムでベクトル計算機を用いると，階数mが小さいときには，かえって遅くなってしまう．

この理由は，①ｄｅＢｏｏｒ－Ｃｏｘのアルゴリズムは漸化式型であるから，ベクトル計算機に適合しない性質をもっていること，②スプライン関数を用いるとき，通常は階数mをあまり大きくしないで，$3 \leqq m \leqq 6$　程度に取るため，最内側ＤＯループ（図１の文番号４０のループ）の長さが小さなものとなること，である．Ｂ－スプラインの階数mを小さくする理由は，もしもそれを

大きくすると，①近似関数（１０）の柔軟さが失われ，近似曲線にうねりを生じやすいこと，② 補間，あてはめなどの問題で，解くべき連立一次方程式の条件が悪くなること，③Ｂ－スプラインの計算量が増大すること，などである．

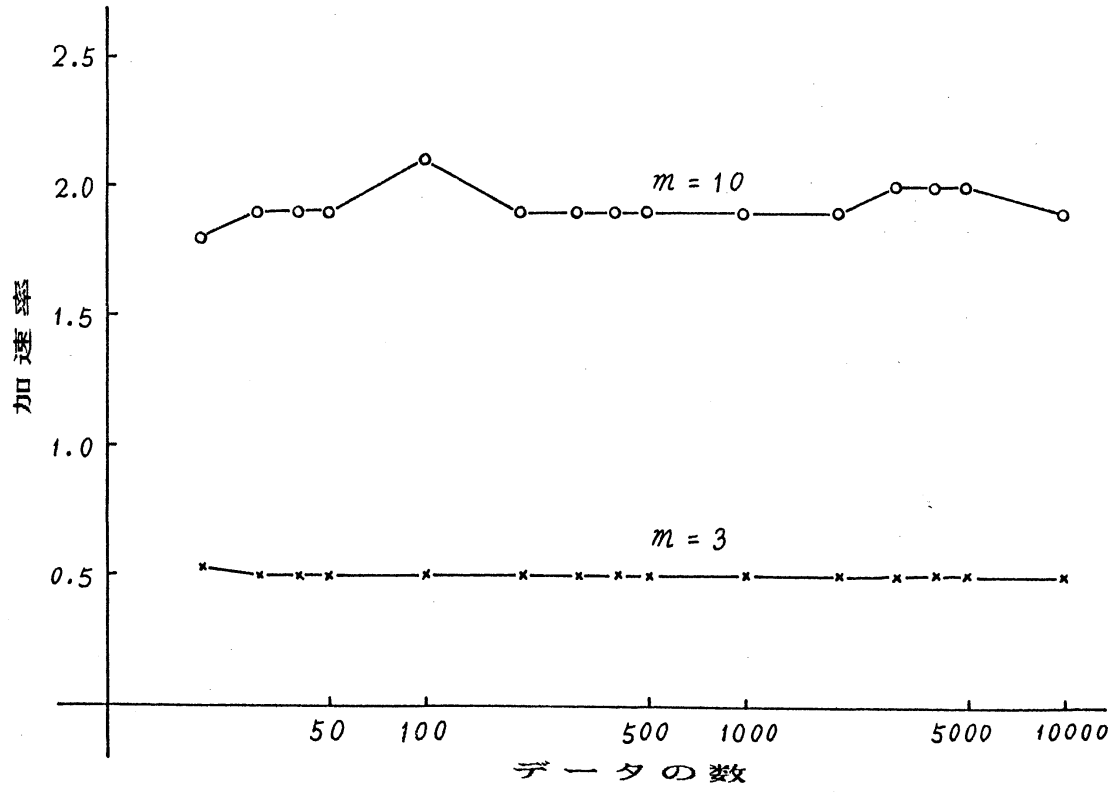

図２　図１のプログラムを用いた場合の加速率（倍精度）

２．２ベクトル計算機向きの計算法

では，どのようにすればベクトル計算機向きの算法になるだろうか？その鍵は近似関数（１０）にある．式（１０）に含まれるＢ－スプライン $N_{m,j}(x)$（$j=1,2,\cdots,4$）は互いに独立であることに注目しよう．これらは，節点が決まれば，すべて決ってしまうものである．したがって，いまＢ－スプラインの値を計算したい点（標本点）x の値 x_k（$k=1,2,\cdots,N$）をあらかじめすべて図１のサブルーチンに与えれば，各標本点でのＢ－スプラインの値を同時に（並列に）計算できるはずである．この点を考慮してプログラムを作ると，最内側ＤＯループの長さを標本点の数 N にすることができる．一般に，高速計算を必要とするような問題では N は大きいと考えてよいので，この方法はベクトル

計算機に適合する．しかし，図１のプログラムのままでこのことを行うと，標本点の数Nがきわめて大きくなるとき，作業領域ＲＭの大きさが無視できなくなる欠点がある．

そこで，ＲＭを用いないで，しかも最内側ＤＯループの長さをNにした計算法ができれば都合がよい．これは，次のようにして実現できる．

式（４），（５），（８）より

$$N_{1,j}(x) = (\xi_j - \xi_{j-1}) M_{1,j}(x) = \begin{cases} 1 & (\xi_{j-1} \leqq x < \xi_j) \\ 0 & (\text{その他}), \end{cases} \qquad (13)$$

$$N_{r,j}(x) = \frac{x - \xi_{j-r}}{\xi_{j-1} - \xi_{j-r}} N_{r-1,j-1}(x) + \frac{\xi_j - x}{\xi_j - \xi_{j-r+1}} N_{r-1,j}(x)$$

$$(r = 2, 3, \cdots, m) \qquad (14)$$

となる．したがって，正規化されたＢ－スプラインだけを用いても（１１）と同様に計算できる．すなわち，$\xi_{j-1} \leqq x < \xi_j$ とするとき，次の配列要素を計算すればよい．

$$\begin{array}{ccccc} & & & & N_{m,j}(x) \\ & & N_{3,j}(x) & & \\ & N_{2,j}(x) & & & N_{m,j+1}(x) \\ N_{1,j}(x) & & N_{3,j+1}(x) & & \vdots \\ & N_{2,j+1}(x) & & & \\ & & N_{3,j+2}(x) & & \\ & & & & N_{m,j+m-1}(x) \end{array} \qquad (15)$$

このとき，計算の順序を，左から右へ，下から上へ，と行うことにすれば，一つの標本点 x に対して大きさmの配列を用意するだけでよい．なぜならば，$N_{r,i}(x)$（$r = 1, 2, \cdots, m$）はrの変化に対して重ね書きできるからである．ここで，$i = j, j+1, \cdots, j+r-1$ である．したがって，いま標本点の数がNであれば，$N \times m$ の２次元配列ＲＮを用意するだけで，ベクトル計算機向

きのアルゴリズムを作成できることになる.

以上の考えに基ずいて作成したプログラムの例が図３である. このプログラムをＭ－３８２とＶＰ－２００にかけ, そのＣＰＵ時間の比を取ると図４のようになる. この図から, 図３のようにプログラミングするとベクトル計算の効果が大きく, データの数が１００以上のとき, 約１０～２５倍の加速率が得られることが分かる.

```
      SUBROUTINE BSPL (X,NOD,NKNOT,GZAI,IOR,IGZ,RN,M6)
C     COMPUTATION OF THE VALUES OF NORMALIZED B-SPLINES
      DIMENSION X(1),GZAI(1),IGZ(1),RN(M6,1)
      REAL*8 X,GZAI,RN
      NOR=NKNOT-IOR
      NORP1=NOR+1
S     DO 20 I=IOR+1,NOR
V     DO 10 KD=1,NOD
V     IF(X(KD) .GE. GZAI(I-1) .AND. X(KD) .LT. GZAI(I)) IGZ(KD)=I
V  10 CONTINUE
S  20 CONTINUE
V     DO 15 KD=1,NOD
V     IF(X(KD) .GE. GZAI(NOR)) IGZ(KD)=NORP1
V  15 CONTINUE
V     DO 30 KD=1,NOD
V     RN(KD,1)=1.0
V  30 CONTINUE
S     DO 40 I=2,IOR
V     DO 50 KD=1,NOD
V     L=IGZ(KD)+I-1
V     K=L-I
V     RN(KD,I)=(X(KD)-GZAI(K))*RN(KD,I-1)/(GZAI(L-1)-GZAI(K))
V  50 CONTINUE
S     DO 60 J=I-1,2,-1
V     DO 70 KD=1,NOD
V     L=IGZ(KD)+J-1
V     K=L-I
V     RN(KD,J)=(X(KD)-GZAI(K))*RN(KD,J-1)/(GZAI(L-1)-GZAI(K))
     *          +(GZAI(L)-X(KD))*RN(KD,J)/(GZAI(L)-GZAI(K+1))
V  70 CONTINUE
S  60 CONTINUE
V     DO 80 KD=1,NOD
V     L=IGZ(KD)
V     K=L-I
V     RN(KD,1)=(GZAI(L)-X(KD))*RN(KD,1)/(GZAI(L)-GZAI(K+1))
V  80 CONTINUE
S  40 CONTINUE
S     DO 90 I=1,IOR
V     DO 100 KD=1,NOD
V     IF(RN(KD,I) .LT. 1.0E-10) RN(KD,I)=0.0
V 100 CONTINUE
S  90 CONTINUE
      RETURN
      END
```

図 3　Ｂ－スプラインの計算プログラム
（ベクトル計算機向きの計算法）

ところで, （１３）, （１４）は階数rに対して漸化式の型をしているので, （１５）のように計算するとき, r の方向（横方向）にはループ・アンローリ

ングを行うことはできない．しかし，$i=j, j+1, \cdots, j+r-1$ の方向（縦方向）にはループ・アンローリングを適用できる．このことを含め，図５は，図３のプログラムで可能な所に二重のループ・アンローリングを行ったものである．このプログラムを用いてＭ－３８２とＶＰ－２００にかけ，そのＣＰＵ時間の比を計算したものが図６である．この図と図４を比較すれば，期待に反し，ループ・アンローリング の効果はほとんどないことが分かる．

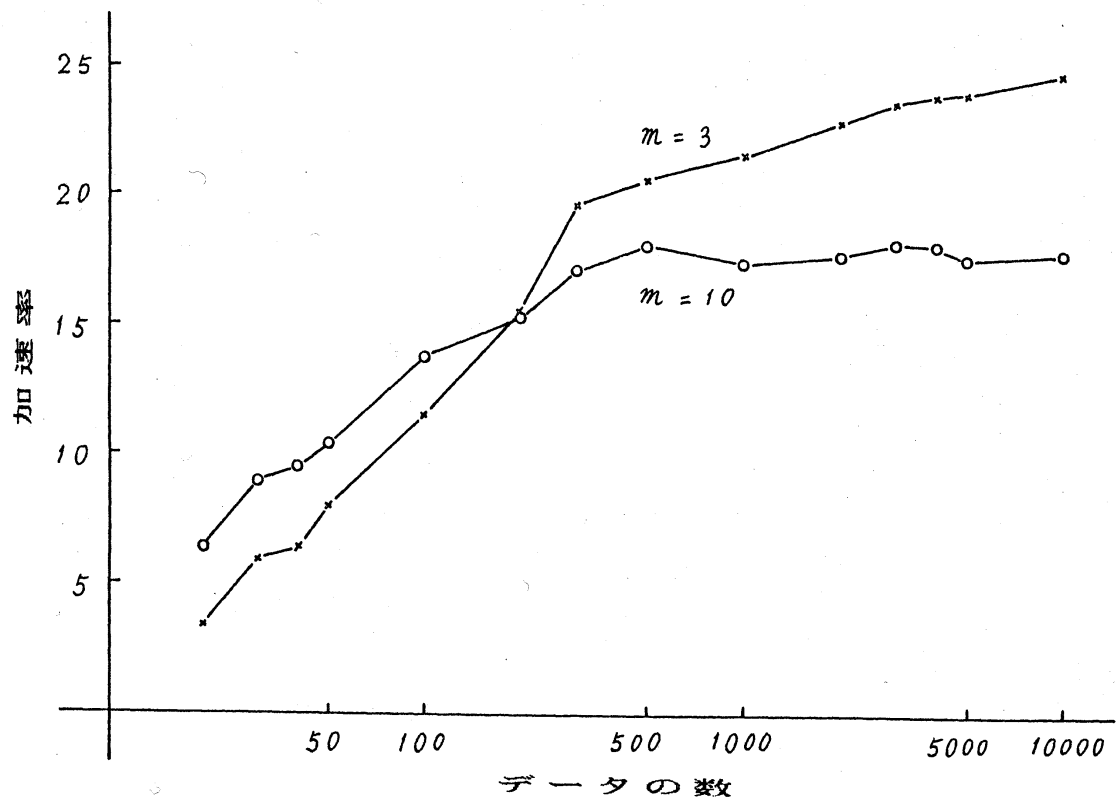

図 4　図３のプログラムを用いた場合の加速率（倍精度）

なお，（４），（５），（１２）を用いた計算法と（１３），（１４）を用いた計算法では，スカラ計算機の場合には前者の方がやや高速であることに注意しておきたい．

```
      SUBROUTINE BSPL (X,NOD,NKNOT,GZAI,IOR,IGZ,RN,M6)
C     COMPUTATION OF THE VALUES OF NORMALIZED B-SPLINES
      DIMENSION X(1),GZAI(1),IGZ(1),RN(M6,1)
      REAL*8 X,GZAI,RN
      NOR=NKNOT-IOR
      NORM1=NOR-1
      NORP1=NOR+1
S     DO 20 I=IOR+1,NORM1,2
V     DO 10 KD=1,NOD
V     IF(X(KD) .GE. GZAI(I-1) .AND. X(KD) .LT. GZAI(I)) IGZ(KD)=I
V     IF(X(KD) .GE. GZAI(I) .AND. X(KD) .LT. GZAI(I+1)) IGZ(KD)=I+1
V  10 CONTINUE
S  20 CONTINUE
      IF(MOD(NKNOT-2*IOR,2) .NE. 0) THEN
V       DO 11 KD=1,NOD
V       IF(X(KD) .GE. GZAI(NORM1) .AND. X(KD) .LT. GZAI(NOR))
     *     IGZ(KD)=NOR
V  11   CONTINUE
      END IF
V     DO 15 KD=1,NOD
V     IF(X(KD) .GE. GZAI(NOR)) IGZ(KD)=NORP1
V  15 CONTINUE
V     DO 30 KD=1,NOD
V     RN(KD,1)=1.0
V  30 CONTINUE
V     DO 40 I=2,IOR
V     DO 50 KD=1,NOD
V     L=IGZ(KD)+I-1
V     K=L-I
V     RN(KD,I)=(X(KD)-GZAI(K))*RN(KD,I-1)/(GZAI(L-1)-GZAI(K))
V  50 CONTINUE
S     DO 60 J=I-1,3,-2
V     DO 70 KD=1,NOD
V     L=IGZ(KD)+J-1
V     K=L-I
V     RN(KD,J)=(X(KD)-GZAI(K))*RN(KD,J-1)/(GZAI(L-1)-GZAI(K))
     *          +(GZAI(L)-X(KD))*RN(KD,J)/(GZAI(L)-GZAI(K+1))
V     RN(KD,J-1)=(X(KD)-GZAI(K-1))*RN(KD,J-2)/(GZAI(L-2)-GZAI(K-1))
     *           +(GZAI(L-1)-X(KD))*RN(KD,J-1)/(GZAI(L-1)-GZAI(K))
V  70 CONTINUE
S  60 CONTINUE
V     IF(MOD(I-2,2) .NE. 0) THEN
V       DO 71 KD=1,NOD
V       L=IGZ(KD)+1
V       K=L-I
V       RN(KD,2)=(X(KD)-GZAI(K))*RN(KD,1)/(GZAI(L-1)-GZAI(K))
     *            +(GZAI(L)-X(KD))*RN(KD,2)/(GZAI(L)-GZAI(K+1))
V  71   CONTINUE
      END IF
V     DO 80 KD=1,NOD
V     L=IGZ(KD)
V     K=L-I
V     RN(KD,1)=(GZAI(L)-X(KD))*RN(KD,1)/(GZAI(L)-GZAI(K+1))
V  80 CONTINUE
V  40 CONTINUE
S     DO 90 I=1,IOR-1,2
V     DO 100 KD=1,NOD
V     IF(RN(KD,I) .LT. 1.0E-10) RN(KD,I)=0.0
V     IF(RN(KD,I+1) .LT. 1.0E-10) RN(KD,I+1)=0.0
V 100 CONTINUE
S  90 CONTINUE
      IF(MOD(IOR,2) .NE. 0) THEN
V       DO 101 KD=1,NOD
V       IF(RN(KD,IOR) .LT. 1.0E-10) RN(KD,IOR)=0.0
V 101   CONTINUE
      END IF
      RETURN
      END
```

図 5　図３のプログラムにループ・アンローリングを施した場合

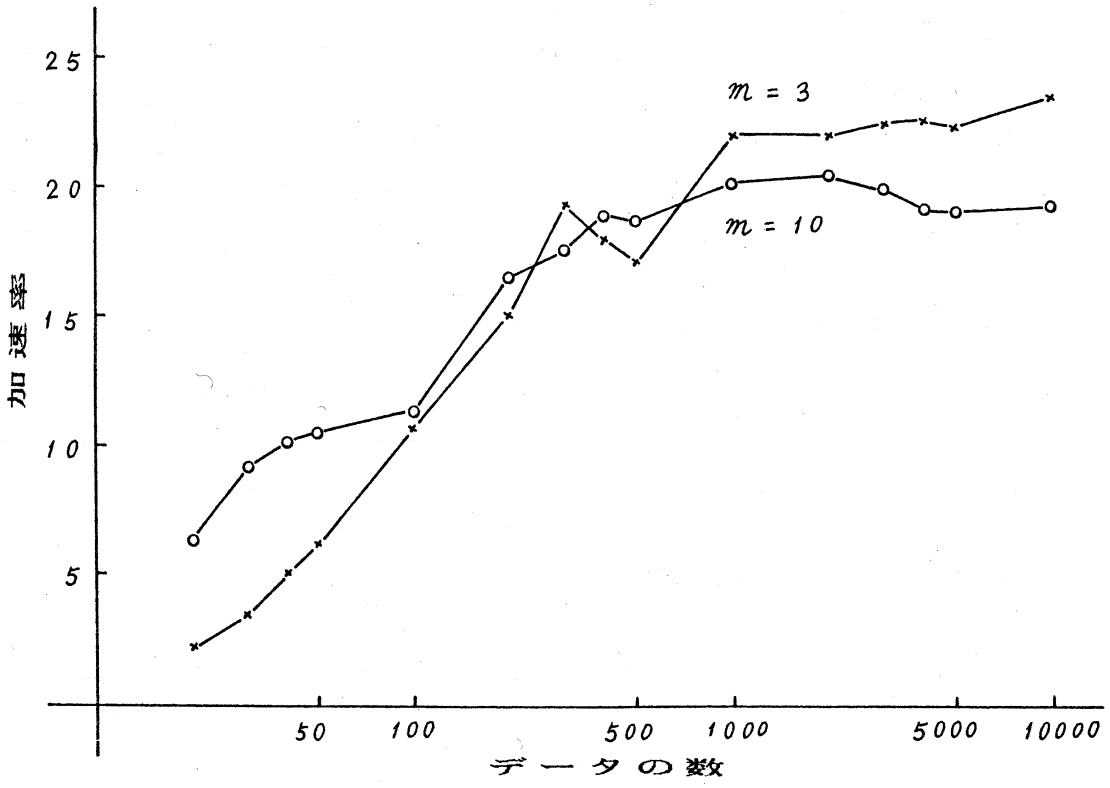

図 6　図5のプログラムを用いた場合の加速率（倍精度）

3．補間・データ平滑化問題への応用

図3または図5のプログラムを用いてB－スプラインの値を計算し，補間および平滑化を行った例を示す．ここでは，1次元データの場合だけについて述べ，2次元以上の場合については別に報告したい．

3．1 補間

区間［a，b］内で，標本点 $x_1, x_2, \cdots, x_N$ （$x_1 < x_2 < \cdots < x_N$）に対して，関数値 $y_1, y_2, \cdots, y_N$ が与えられているものとする．このデータを $m-1$ 次のスプライン関数 $S(x)$ を用いて補間することを考えよう．いま，$2m$ 個の付加節点を区間［a，b］の両端に置き，

$$\left.\begin{aligned} \xi_{1-m} &= \cdots = \xi_{-1} = \xi_0 = a \\ \xi_{h-m+1} &= \cdots = \xi_{h-1} = \xi_h = b \end{aligned}\right\} \qquad (16)$$

とする．また，内部の節点を

$$\xi_i = x_{i+[m/2]} \qquad (i = 1, 2, \cdots, h-m) \qquad (17)$$

とする．これら内部の節点は必ずしも標本点と一致させる必要はなく，節点を変化させていろいろな補間を求め，その中からよいものを選ぶこともできる[2)]．しかし，ここでは簡単なため，（１７）のようにする．

このとき，節点 ξ_{i-m} , $\xi_{i-m+1}, \cdots, \xi_i$ に対して定義された m 階のＢ－スプライン $N_{m\,i}(x)$（$i = 1, 2, \cdots, h$ ）を用いて，

$$S(x) = \sum_{i=1}^{h} c_i N_{m\,i}(x) \qquad (18)$$

と表すことができる．この式が，与えられたデータ点を通ることから，

$$\sum_{i=1}^{h} c_i N_{m\,i}(x_k) = y_k \qquad (k = 1, 2, \cdots, N) \qquad (19)$$

を得る．ここで $h = N$ である．式（１９）は c_i（$i = 1, 2, \cdots, h$ ）を未知数とする連立一次方程式であり，その係数行列は，Ｂ－スプラインの局所性（６）から，帯の幅が高々 $2m-1$ の帯行列になる．式（１９）を解いて，その解を（１８）へ代入すれば，区間［a , b ］内の任意の点 x で補間値を求めることができる．

したがって，ある節点の組に対するスプライン関数を用いた補間では，主として次の５個の部分の計算が必要である．①すべての標本点でのＢ－スプラインの値の計算，②式（１９）の要素の決定，③式（１９）の解 c_i（$i = 1, 2, \cdots, h$ ）の計算，④補間値を計算したいすべての点でのＢ－スプラインの値の計算，⑤式（１８）による補間値の計算．このうちで多くの時間を消費するのは，通常①，③，④である．この中で①，④は２．２で述べた方法で高速に計算できる．また，データをあらかじめすべて与えてベクトル長を長くする考え方は，⑤にも適用できる．

さて，Ｂ－スプラインの計算に図３のプログラムを用いた場合の，計算結果を示すことにしよう．表１は，区間［０　，５］で等間隔に与えられた１０１個のデータを補間し，等間隔な５０１個の点で補間値を計算した場合の，各部分の計算時間を示している．用いた計算機はＭ－３８２である．この表からも，

表 1　補間計算の各部分の所要時間（単精度，単位ｍｓ）

計算部分 / 階数m	①標本点でのＢ－スプラインの値の計算	②式（１９）の要素の決定	③式（１９）の解 c_i の計算	④補間点でのＢ－スプラインの値の計算	⑤式（１８）による補間値の計算	①～⑤の全体
3	9．7	7．5	18．3	48．8	2．3	86．6
4	12．6	7．4	17．8	62．1	2．8	102．7
10	23．0	8．2	21．2	115．2	6．6	174．2

Ｂ－スプラインの値の計算を高速化すれば，全体の高速化に大きく貢献することが分かる．

図７は，同じデータをＭ－３８２とＶＰ－２００を用いて補間したときの，ＣＰＵ時間の比である．階数（次数＋１）m の上昇とともに少し下っているが，１０倍前後の加速率が得られている．

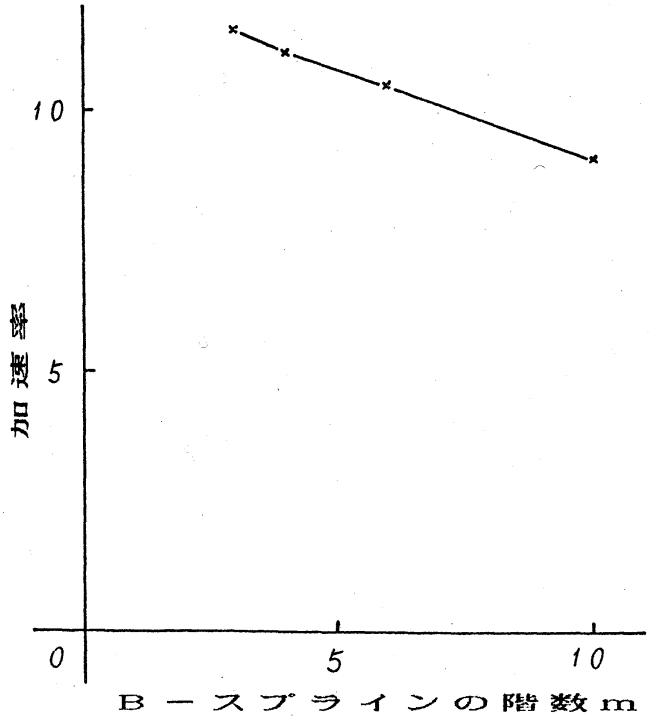

図 7　補間計算の加速率（単精度）
（データ数１０１，補間点５０１）

なお，ＦＡＣＯＭ　ＳＳＬⅡの中のＢＩＣ３およびＢＩＦ３を用いて同じ補間を行ったとき，５次のスプライン関数（$m=6$）で約１．３倍の加速率であった．

３．２データ平滑化

区間［a，b］内で，標本点 $x_1, x_2, \cdots, x_N$ に対して，データ

$$F_k = f(x_k) + \varepsilon_k \quad (k=1, 2, \cdots, N) \tag{20}$$

が与えられているものとする．ここで，$f(x)$ は未知の関数（信号）であり，ε_k は測定誤差である．

データ（２０）の近似関数として，$m-1$次のスプライン関数 $S(x)$ を用いる．補間のときと同様に，$2m$ 個の付加節点を区間の両端に置き，内部の節点を $\xi_1 \leqq \xi_2 \leqq \cdots \leqq \xi_{n-m}$ とすると，（１８）と同様に

$$S(x) = \sum_{i=1}^{n} c_i N_{m,i}(x) \tag{21}$$

と表すことができる．ここで，内部の節点はm個まで重ねることができる．

最小二乗法を用いて（２１）を（２０）へあてはめることにより，データ平滑化を行う．このとき，適切な節点の数と位置を決めることが，良い近似を得る鍵となるが，ここでは，それはあらかじめ決められているものとして，パラメタ c_i（$i=1, 2, \cdots, n$）のみを未知数とする線形最小二乗法の問題を扱う．残差の二乗和は，

$$Q = \sum_{k=1}^{N} w_k \left\{ S(x_k) - F_k \right\}^2 \tag{22}$$

となる．ここで，w_k はデータの重みである．

式（２２）をパラメタ c_i（$i=1, 2, \cdots, n$）で偏微分して零とおくと，正規方程式

$$A\,\boldsymbol{c} = \boldsymbol{d} \tag{23}$$

を得る．ここで，

$$\boldsymbol{c} = (c_1, c_2, \cdots, c_n)^T \tag{24}$$

である．式（２３）の係数行列Aは，Ｂ－スプラインの局所性（６）から，帯の幅が $2m-1$ の対称な帯行列になる．また，（２３）は，次数 $m-1$ があまり高くないかぎり条件のよい方程式であるので，たとえばコレスキー法を用いて解くことができる．その解 c_i（$i=1, 2, \cdots, n$）を（２１）へ代入すれば，区間［a，b］内の任意の点xで平滑値を求めることができる．

さて，良い近似を得るためには，節点の数と位置をいろいろ変えてあてはめを計算し，その中から良いものを選び出す必要がある．このためには，あてはめの良さを評価する規準がなければならない．この規準としてよく用いられるものに，誤差分散の不偏推定量δと赤池の情報量規準 AIC がある．いま，回帰模型を

$$F_k = S(x_k) + \varepsilon_k \quad (k=1, 2, \cdots, N) \qquad (25)$$

とする．ここで，誤差 ε_k は互に独立で，平均値 0，分散 σ^2 の正規分布に従うものと仮定する．このとき，誤差分散の不偏推定量は

$$\delta = Q/(N-n) \qquad (26)$$

となり，赤池の情報量規準は

$$AIC = N \log_e Q + 2n \qquad (27)$$

と表すことができる．ここで，Q は（２２）であり，n はパラメタ c_i（$i=1, 2, \cdots, n$）の数である．

以上のことから，ある節点の組に対するあてはめ（１回分のあてはめ）において，主に次の６個の部分の計算が必要なことになる．①すべての標本点でのＢ－スプラインの値の計算，②式（２３）の要素の計算，③式（２３）の解c_i（$i=1, 2, \cdots, n$）の計算，④残差，あてはめの規準の計算，⑤平滑値（近似関数値）を計算したいすべての点でのＢ－スプラインの値の計算，⑥式（２１）による平滑値の計算．

このうちで多くの時間を消費するのは，通常①，②，⑤であるが，①，⑤は２．２で述べたようにして高速に計算できる．また，②，④，⑥も，データをあらかじめすべて与えることによりベクトル長を長くできるので，高速な計算が可能である．

では，Ｂ－スプラインの計算に図５のプログラムを用いた場合の，計算結果を示すことにしよう．表２は，区間［０，１］で与えられた４００個のデータ

表２　平滑化計算の各部分の所要時間（単精度，単位ｍｓ）

計算部分 / 階数m	①標本点でのＢ－スプラインの値の計算	②式（２３）の要素の計算	③式（２３）の解 c_i の計算	④残差，あてはめの規準の計算	⑤平滑化点でのＢ－スプラインの値の計算	⑥式（２１）による平滑値の計算	①～⑥の全体
3	10.0	4.2	0.3	2.2	9.1	1.9	27.7
4	13.4	6.3	0.4	2.8	13.5	2.4	38.8
10	65.9	28.5	1.4	5.8	65.6	5.8	173.0

を平滑化し，等間隔な４０１個の点で平滑値を計算した場合の各部分の計算時間を示している．ここで，内部の節点の数は９であり，用いた計算機はＭ－３８２である．この表からも，Ｂ－スプラインの計算の高速化の意義が分かる．

図８は，同じデータをＭ－３８２とＶＰ－２００を用いて平滑化したときのＣＰＵ時間の比である．この場合にも，補間と同様にスプライン関数の次数にあまり影響されず，約９倍の加速率が得られている．

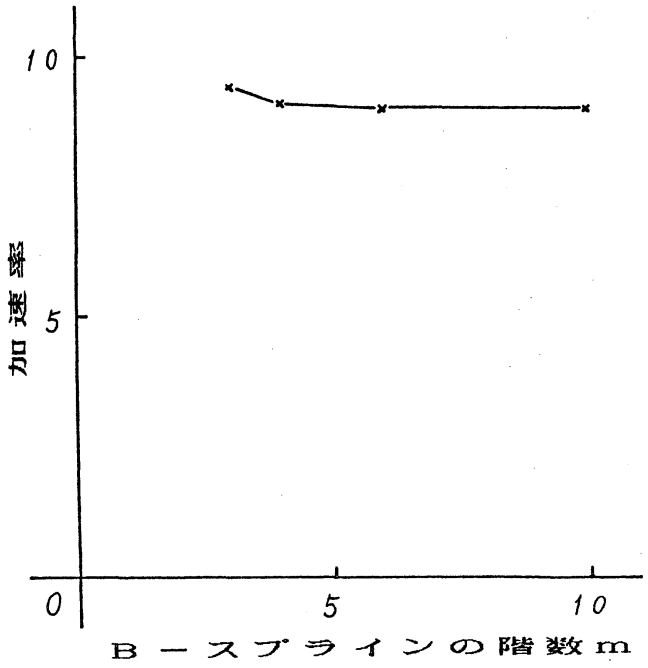

図８　平滑化計算の加速率（単精度）
（データ数４００，平滑化点４０１）

なお，ＦＡＣＯＭ　ＳＳＬⅡの中のＢＳＣ１およびＢＳＦ１を用いて同じデータを平滑化したとき，５次のスプライン関数（$m=6$）で内部の節点が９個のとき，約１．２倍の加速率であった．

４．むすび

本文では，ベクトル計算機向きのＢ－スプラインの計算法と，その補間，データ平滑化問題への応用について述べた．Ｂ－スプラインの値を計算するためのｄｅ　ＢｏｏｒーＣｏｘのアルゴリズムは，漸化式型であるため，標本点一つずつに対して逐次的にＢ－スプラインの値を計算する方法では，ベクトル計算機に適合しない．しかし，近似関数に含まれるＢ－スプラインが互いに独立であることに着目し，あらかじめすべての標本点を与えてＢ－スプラインの値を計算することにすれば，ベクトル計算機に適した算法となることが明らかになった．ここで述べた方法を用いれば，１次元の補間，あてはめにおいて１０倍程度の加速率を達成できる．

なお，本研究の一部は京都大学大型計算機センターの開発課題として行われている．

参　考　文　献

１）市田，吉本：スプライン関数とその応用，教育出版，東京，１９７９．
２）吉本：自由節点のスプライン関数を用いた補間について，情報処理学会論文誌，Ｖｏｌ．２２，ｐｐ．３０４－３１１，１９８１．