CREATIVE MATH.
14 (2005), 75 - 82

On the spline and B-spline approximation of a function

Magnolia Rebeleş

Abstract. The functions are main mathematical main tools in describing and analyzing physical processes and phenomena in nature. Only in a few cases the functions modeling these phenomena are known explicitly, in the most of the cases there are used approximations.
In this paper the power of the spline and B-spline approximation for a given function is presented. A code algorithm generates a table in which the values of these two approximations can be compared. The purpose of this utility programme is to be a didactic tool on how the B-spline functions perform a better approximation than the spline functions.

1. Introduction

Let

$$\Delta := \{x_i\}_1^N, a = x_0 < x_1 < ... < x_N < x_{N+1} = b$$

be a partition of $[a, b]$ into N subintervals $I_i = [x_i, x_{i+1}[, i = \overline{0, N-1}$ and $I_N = [x_N, x_{N+1}]$.

Let m be a positive integer and let $M = (m_1, m_2, ..., m_N)$ be a vector with integer components satisfying $1 \leq m_i \leq m, i = \overline{1, N}$ and $k := \sum_{i=1}^N m_i$.

For the linear space of polynomial spline functions of degree $m-1$ with multiplicity vector M with respects to the partition Δ

$$S(P_m, M, \Delta) = \{s : [a, b] \to R/\exists s_0, s_1, ..., s_N \in P_m :$$

$$s(x) = s_i(x), x \in I_i, i = \overline{0, N},$$

$$D^j s_{i-1}(x_i) = D^j s_i(x_i), j = \overline{0, m-1-m_i}, i = \overline{1, N}\}$$

(P_m is the set of the polynomials of degree at most $m-1$) let us consider two bases B_1 and B_2:

$$B_1 = \{(x - x_i)_+^{m-j}\}, i = \overline{0, N}, j = \overline{1, m_i}, m_0 := m$$

and

$$B_2 = \{N_{i,m}(x) | i = \overline{1, d}, x \in [a, b]\}$$

where $d = m + k$ and

$$N_{i,m}(x) := \begin{cases} (-1)^m (t_{i+m} - t_i)[t_i, ..., t_{i+m}](x - t)_+^{m-1}, & t_i < t_{i+m} \\ 0, & \text{otherwise} \end{cases}$$

Received: 18.04.2005. In revised form: 02.05.2005
2000 *Mathematics Subject Classification.* 41A15.
Key words and phrases. *Spline interpolation, B-spline functions.*

are the normalized B-spline functions defined as the divided differences of the function $(x-t)_+^{m-1}$ at the nodes $t_1, t_2, ..., t_{i+m}$ of the partition

$$\Omega : t_1 \leq t_2 \leq ... \leq t_{2m+k}, \tag{1.1}$$

such that

$$t_1 \leq ... \leq t_m \leq a, b \leq t_{m+k+1} \leq ... \leq t_{2m+k}$$

and

$$t_{m+1} \leq ... \leq t_{m+k} = \underbrace{x_1, x_1, ..., x_1}_{m_1}, ..., \underbrace{x_N, x_N, ..., x_N}_{m_N},$$

see [2]

The basis B_1 of $S(P_m, \Delta, M)$ is very useful from a theoretical point of view. For applications the use of this basis is not always beneficial because there is a small precision on approximation of a function with the linear combinations of functions from B_1

$$s(x) = \sum_{i=0}^{N} \sum_{j=1}^{m_i} c_{i,j}(x - x_i)_+^{m-j}.$$

The basis B_2 avoids this disadvantage and it is also easier to use in the applications.

The algorithm presented in the next section uses the following three results.

Theorem 1.1. *Let $S_m(\Delta)$ be the space of the spline functions of degree $m-1$ with the simple nodes $x_1, ..., x_N$ ($m_i = 1$ for all $i = \overline{1, N}$, $S_m(\Delta) \subset S(P_m, \Delta, M)$). Let $f \in H^{m,2} := \{f \in C^{m-1}[a,b] | f^{(m-1)}$ absolute continuous on $[a,b]$ and $f^{(m)} \in L^2[a,b]\}$ be a function with the given real numbers $f(x_i), i = \overline{0, N+1}$, $x_i \in \Delta$. The function*

$$Sf(x) = \sum_{i=0}^{N+1} s_i(x) f(x_i) \in S_m(\Delta) \tag{1.2}$$

interpolates the function f at the nodes x_i.
The functions

$$s_i(x) = \sum_{j=0}^{m-1} a_j x^j + \sum_{i=0}^{N+1} b_i (x - x_i)_+^{2m-1}$$

are called the fundamental natural spline functions. The coefficients $a_j, b_i, j = \overline{0, m-1}, i = \overline{0, N+1}$ are determined from the spline conditions

$$\begin{cases} s_i(x_j) = \delta_{ij}, & i, j = \overline{0, N+1} \\ s_i^{(p)}(x) = 0, & p = \overline{m, 2m-1}, x \in (x_N, b] \end{cases}$$

Theorem 1.2. *If $\{N_i, m\}_1^{m+k}$ are normalized B-splines associated with the extended partition Ω (1.1), then every $s \in S(P_m, M, \Delta)$ has a unique expansion of the form*

$$Bf(t) := s(t) = \sum_{i=1}^{m+k} f(t_i) N_{i,m}(t), \forall t_m \leq t < t_{m+k+1} \tag{1.3}$$

called the B-spline expansion of s. [3]

Theorem 1.3. *If $m \geq 2$ is an integer and Ω is the partition (1.1), then the recurrence relation of B-spline functions holds*

$$M_{i,m}(t) = \frac{t - t_i}{t_{i+m} - t_i} M_{i,m-1}(t) + \frac{t_{i+m} - t}{t_{i+m} - t_i} M_{i+1,m-1}(t), \forall t \in [a, b] \tag{1.4}$$

where

$$M_{i,m} = \frac{N_{i,m}}{(t_{i+m} - t_i)}.$$

2. Main result

There are known Matlab routines which generate the values of the given numbers $f(x_i)$, the function f being unknown. In order to observe the "good approximation" of the B-spline functions comparing with the spline function approximation, it is necessary to know the approximated function f.

Problem 2.1. *To implement a routine which generate the values of a function f, the values of its associated spline function Sf and the values of the B-spline expansion in order to estimate the best approximation.*

Step 1: The computation of the B-spline functions $M_{i,m}$ using the recurrence relation (1.4)

$$M_{i,m}(t) = \frac{t - t_i}{t_{i+m} - t_i} M_{i,m-1}(t) + \frac{t_{i+m} - t}{t_{i+m} - t_i} M_{i+1,m-1}(t), \forall t \in [a, b];$$

Step 2: The computation of the normalized B-spline functions $N_{i,m}$

$$N_{i,m} = (t_{i+m} - t_i) M_{i,m}, i = \overline{1, N};$$

Step 3: The computation of the B-spline expansion of s at a given $t \in [a, b]$

$$Bf(t) := s(t) = \sum_{i=1}^{m+k} f(t_i) N_{i,m}(t);$$

Step 4: The computation of the spline function at a given $x \in [a, b]$

$$Sf(x) = \sum_{i=1}^{N} s_i(x) f(x_i);$$

Step 5: The completion of the first row of the table with the above computed values $f(t_i), t_i \in [a, b], h = 0.1$;

Step 6: The completion of the second row of the table with the spline values $Sf(t_i)$ at a fairly large number of $t_i \in [a, b], h = 0.1$;

Step 7: The completion of the third row of the table with the B-spline expansion $Bf(t_i)$ at a fairly large number of $t_i \in [a, b], h = 0.1$.

- Input:

f a function which may be chosen from the set of the functions

$$\{sin(x), cos(x), e^x, \frac{1}{x+1}\}$$

$T = \{t_1, t_2, ..., t_N\}$ the extended partition with z equal values at the both extremes of the interval

$$a = t_1 = ... = t_z < \underbrace{t_{z+1} < ... < t_{N-z}}_{p\ nodes} < t_{N-z+1} = ... = t_N = b$$

$$Y = \{\underbrace{t_1, y_1, y_2, ..., y_n, t_N}_{q\ nodes}\}, q \leq p$$

the vector of the control points of the spline approximation

$$X = \{x_1, x_2, ..., x_z\}$$

the vector of the control points of the B-spline approximation m the order of the spline and B-spline function, $m = z$.

- Output:

$f(t)$ the f values of the function f computed on the equal spaced points in $[t_1, t_N]$ with the step size $h = 0.1$
Sf the values of the spline function associated to f
Bf the values of the B-spline function associated to f

The algorithm is based on the Schumaker's triangular array [3] that generates the values $N_{i,m}, i = \overline{1, d}, t \in [t_m, t_{d+1}]$. Some conditions are imposed during the code algorithm in order to obtain well dimensioned matrices. The function we implemented in Matlab language

$$[tabelf] = final(m, T, Y, X, functia)$$

generates the three values arrays f, Sf, Bf.

Application 2.1. *To illustrate the applicability of the above problem, let us consider the following given data:*
-the function

$$f : [1, 2] \to \boldsymbol{R}, f(x) = \frac{1}{x+1}$$

-the extended partition $T = [1\ \ 1\ \ 1\ \ 1.3\ \ 1.4\ \ 1.5\ \ 1.7\ \ 2\ \ 2\ \ 2]$
-the vector of the control points of the spline function $Y = [1\ \ 1.5\ \ 1.7\ \ 2]$
-the vector of the control points of the B-spline function $X = [1.3\ \ 1.4\ \ 1.7]$
-the order of the spline and B-spline function $m = 3$

The function

$$[tabelf] = final(3, [1\ \ 1\ \ 1\ \ 1.3\ \ 1.4\ \ 1.5\ \ 1.7\ \ 2\ \ 2\ \ 2], [1\ \ 1.5\ \ 1.7\ \ 2], [1.3\ \ 1.4\ \ 1.7], '1/(x+1)')$$

returns the values

t	1	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9
f	0.5000	0.4762	0.4545	0.4348	0.4167	0.4000	0.3846	0.3704	0.3571	0.3448
Sf	0.5000	0.4767	0.4551	0.4351	0.4168	0.4000	0.3846	0.3704	0.3572	0.3449
Bf	0	0.0483	0.1932	0.4348	0.4167	0.3748	0.4949	0.3704	0.1646	0.0412

Remark 2.1. *The B-spline expansion offers etter approximation of the function f but only on the control points X. For instance,*

$$f(1.4) = Bf(1.4) < Sf(1.4).$$

Remark 2.2. *For the equal vectors of the control points $X = Y$, or equal control points $x_i = y_i$, the approximation*

$$f(1.7) = Sf(1.7) = Bf(1.7)$$

is obtained.

3. The code algorithm

Algorithm 3.1. *(The construct of the table with the required values)*

```
function [tabelf]=final(m,T,Y,X,functia);
% m=the order of the spline and B-spline functions
% T=the extended partition
% Y=the control points' vector of the spline function
% X=the control points' vector of the B-spline function
N=length(T); n=length(X);
suma=0;
for i = T(1,1) : .1 : T(1,N)
.       suma = suma + 1;
end;
k = T(1,1) : .1 : T(1,N);
switch (functia)
case '1/(x+1)'
.       for i = 1 : suma - 1
.            tabelf(1,i) = 1/(1+k(i)); %the values of the chosen function f
.       end;
.       tabelf(2,:) = tabel_spline(m,Y,functia); %the values of the spline Sf
.       tabelf(3,:) = tabelB_spline(m,T,X,functia); %the values of the B-spline expansion
break;
case 'sin(x)'
.       for i = 1 : suma - 1
.            tabelf(1,i) = sin(k(i));
.       end;
.       tabelf(2,:) = tabel_spline(m,Y,functia);
.       tabelf(3,:) = tabelB_spline(m,T,X,functia);
break;
case 'cos(x)'
.       for i = 1 : suma - 1
.            tabelf(1,i) = cos(k(i));
.       end;
.       tabelf(2,:) = tabel_spline(m,Y,functia);
.       tabelf(3,:) = tabelB_spline(m,T,X,functia);
break;
case 'exp(x)'
.       for i = 1 : suma - 1
.            tabelf(1,i) = exp(k(i));
end;
.       tabelf(2,:) = tabel_spline(m,Y,functia);
.       tabelf(3,:) = tabelB_spline(m,T,X,functia);
break;
end;
```

Routine 3.1. *(The computation of the spline function $Sf(t)$,$t \in [a,b], h = 0.1$)*

```
function [tabel2]=tabel_spline(m,Y,functia)
n=length(Y);
suma=0;
for i = Y(1,1) : .1 : Y(1,n)
.       suma = suma + 1;
end;
k = Y(1,1) : .1 : Y(1,n);
for i = 1 : suma - 1
```

```
.       tabel2(i)=aprox_cu_spline(m,Y,k(i),functia);
end;
```

Routine 3.2. *(The computation of the B-spline expansion $Bf(t)$, $t \in [a,b], h = 0.1$)*

```
function [sir]=tabelB_spline(m,T,X,functia)
N=length(T); n=length(X);
sum=0;
for i = T(1,1) : .1 : T(1,N)
.       sum = sum + 1;
end;
k = T(1,1) : .1 : T(1,N);
for i = 1 : sum - 1
.       sir(i)=aprox_cu_B_spline(m,T,X,k(i),functia);
end;
```

Routine 3.3. *(The computation of the spline function $Sf(t)$, $t \in Y$)*

```
function [aps]=aprox_cu_spline(m,Y,x,functia)
N=length(Y); A=zeros(N+m,N+m);
for i=1:N
.       A(i,1)=1;
end;
for i=1:N
.       for j=2:m
.               A(i,j) = (Y(1,i))^(j-1);
.       end;
end;
for i=1:N
.       for j=m+1:m+N
.               A(i,j) = (fplus(Y(1,i) - Y(1,j-m)))^(2*m-1);
.       end;
end;
for i=N+1:N+m
.       for j=m+1:m+N
.               A(i,j) = ((facto(2*m-1))/(facto(2*m-1-i+2))) * (((Y(1,N)+1) - Y(1,j-m))^(2*m-1-i+2));
.       end;
end;
B=zeros(N+m,1);
switch (functia) case '1/(x+1)'
.       for i=1:N
.               B(i,1)=1/(1+Y(1,i));
.                       for j=N+1:N+m
.                               B(j,1) = ((-1)^(j-N+m-1)) * facto(j-N+m-1) * ((1+(Y(1,N)+1))^(-j+N-m));
.                       end;
.       end;
.       R=A \ B;
.       C=zeros(1,N+m);
.       C(1,1)=1;
.       for i=2:m
.               C(1,i) = x^(i-1);
.       end;
.       for i=m+1:N+m-1
.               C(1,i) = (fplus((x - Y(1,i-m))))^(2*m-1);
.       end;
.       aps = C * R;
break;
case 'sin(x)'
.       for i=1:N
.               B(i,1)=sin(Y(1,i));
.                       for j=N+1:N+m
.                               B(j,1) = sin(Y(1,N) + 1 + ((j-N+m-1)*pi)/2);
.                       end;
.       end;
.       R=A\ B;
.       C=zeros(1,N+m);
.       C(1,1)=1;
.       for i=2:m
.               C(1,i) = x^(i-1);
.       end;
.       for i=m+1:N+m-1
.               C(1,i) = (fplus((x - Y(1,i-m))))^(2*m-1);
.       end;
.       aps = C * R;
break;
```

```
case 'cos(x)'
.       for i=1:N
.            B(i,1)=cos(Y(1,i));
.                for j=N+1:N+m
.                    B(j,1) = cos(Y(1,N) + 1 + ((j - N + m - 1) * pi)/2);
.                end;
.       end;
.       R=A\ B;
.       C=zeros(1,N+m);
.       C(1,1)=1;
.       for i=2:m
.       C(1,i) = x^(i-1);
.       end;
.       for i=m+1:N+m-1
.            C(1,i) = (fplus((x - Y(1,i - m))))^(2*m-1);
.       end;
.       aps = C * R;
break;
case 'exp(x)'
.       for i=1:N
.            B(i,1)=exp(Y(1,i));
.                for j=N+1:N+m
.                    B(j,1)=exp(j-N+m-1);
.                end;
.       end;
.       R=A\ B;
.       C=zeros(1,N+m);
.       C(1,1)=1;
.       for i=2:m
.            C(1,i) = x^(i-1);
.       end;
.       for i=m+1:N+m-1
.            C(1,i) = (fplus((x - Y(1,i - m))))^(2*m-1);
.       end;
.       aps = C * R; break;
```

Routine 3.4. *(The computation of the spline function $Bf(t)$, $t \in X$)*

```
function [apbs]=aprox_cu_B_spline(m,T,X,t,functia)
N=length(T); n=length(X); A=zeros(m,N);
for i=1:m
.       for j=1:N
.            A(i,:)=B_spline1(m,T,X(1,i));
.       end;
end;
F=zeros(n,1);
switch (functia) case '1/(x+1)'
.       for i=1:n
.            F(i,1)=1/(1+X(1,i));
.       end;
.       C=A\ F;
.       apbs = B_spline1(m,T,t) * C;
break;
case 'sin(x)'
.       for i=1:n
.            F(i,1)=sin(X(1,i));
.       end;
.       C=A\ F;
.       apbs=B_spline1(m,T,t)*C;
break;
case 'cos(x)'
.       for i=1:n
.            F(i,1)=cos(X(1,i));
.       end;
.       C=A\ F;
.       apbs = B_spline1(m,T,t) * C;
break;
case 'exp(x)'
.       for i=1:n
.            F(i,1)=exp(X(1,i));
.       end;
.       C=A\ F;
.       apbs = B_spline1(m,T,t) * C;
break;
end;
```

Routine 3.5. *function [B_spline_normate]=B_spline1(m,T,t)*

```
d=length(T); M=zeros(m,d);
for j=1:d-1
.       if ((T(1, j) < T(1, j + 1))&(t >= T(1, j))&(t < T(1, j + 1)))
.            M(1, j) = 1/(T(1, j + 1) − T(1, j));
.       end;
end;
for i=2:m
.       for j=1:d-1
.            if ((i + j <= d)&(T(1, i + j) > T(1, j)))
.                 M(i, j) = ((t − T(1, j)) ∗ M(i − 1, j) + (T(1, i + j) − t) ∗ M(i − 1, j + 1))/(T(1, i + j) − T(1, j));
.            end;
.       end;
end;
No=zeros(m,d);
for j=1:d-1
.       if ((t >= T(1, j))&(t < T(1, j + 1)))
.            No(1,j)=1;
.       end;
end;
for i=2:m
.       for j=1:d-1
.            if (i + j <= d)
.                 No(i, j) = (t − T(1, j)) ∗ M(i − 1, j) + (T(1, i + j) − t) ∗ M(i − 1, j + 1);
.            end;
.       end;
end;
for i=1:d
.       B_spline_normate(i)=No(m,i);
end;
```

References

[1] Goodman T.N.T., Lee S.L., *Spline approximation operators of Bernstein-Schoenberg type in one and two variables*, J. Approx. Theory 33 (1982), 248-263

[2] Micula G., Micula S., *Handbook of Splines*, Kluwer Academic Publishers, Dordrecht/Boston/London, 1999

[3] Schumaker L., *Spline functions. Basic theory*, A Wiley-Interscience Publication, John Wiley & Sons, New York, 1980

[4] Stancu D.D., Coman Gh., Blaga P., *Analiză numerică şi teoria aproximării*, Presa Universitară Clujeană, Cluj-Napoca, 2002

Colegiul Economic Nicolae Titulescu
Progresului 45
Baia Mare 430291, Romania
E-mail address: m_rebeles@yahoo.com