Derivation of Cubic Splines from Cubic Hermite Functions

Saulo Pomponet Oliveira [1]
saulo@ime.unicamp.br
Prof. Dr. Petrônio Pulino [2]
pulino@ime.unicamp.br

DMA - IMECC - UNICAMP
Caixa Postal 6065, Campinas-SP
Brazil 13081-970

Abstract: There are several ways to derive the cubic splines. In this article we derive them using piecewise cubic hermite functions with a very simple approach: we put continuity conditions on the second derivative of an arbitrary function $s = s(x)$ in the linear space spanned by the hermite functions. This way, s will also belong to the piecewise cubic splines space. It reveals a nice connection between splines and hermite functions that can be explored in numerical analysis courses.

[1] Student of the M.Sc. program in Applied Mathematics, IMECC-UNICAMP
[2] Department of Applied Mathematics, IMECC-UNICAMP

1 Piecewise Polynomials

Let $\pi : a = x_0 < x_1 < \ldots < x_n = b$ be a partition of the interval $[a, b] \subset I\!R$. It means that the $n + 1$ points of π divide $[a, b]$ in n subintervals $[x_k, x_{k+1}]$, $0 \leq k \leq n - 1$. We denote an arbitrary interval $[x_k, x_{k+1}]$ by K. Let $P_m([x_k, x_{k+1}]) = P_m(K)$ be the linear space of the polynomials with degree $\leq m$ defined in K. We call the set

$$PP_m(\pi) = \{\ q : [a, b] \to I\!R \mid \forall k \ \ , \ \ 0 \leq k \leq n - 1,$$
$$\exists p_{(k)} \in P_m(K)\ ;\ p_{(k)}(x) = q(x)\ \forall x \in K\ \} \tag{1}$$

as the piecewise polynomials space with degree $\leq m$ defined in π. $PP_m(\pi)$ is a linear space with dimension $(m+1)n$, and $P_m([a, b])$ is a subset of $PP_m(\pi)$.

When we put interpolatory or smoothness conditions in $PP_m(\pi)$, we generate other spaces that are subspaces of $PP_m(\pi)$. We will consider two subspaces of $PP_3(\pi)$, the cubic splines and cubic hermite functions spaces.

2 $\mathrm{S}_3(\pi)$ Space

Given a partition π of [a,b], $\mathrm{S}_3(\pi)$ is the set of all fuctions $s \in C^2[a, b]$ that reduces to cubic polynomials at each interval $[x_k, x_{k+1}]$, $0 \leq k \leq n - 1$. These functions are called cubic splines. We can also define $\mathrm{S}_3(\pi)$ as

$$\mathrm{S}_3(\pi) = PP_3(\pi) \cap C^2[a, b] \tag{2}$$

Since splines are twice continuously differentiable, they're suitable to approximation problems, especially this interpolation problem: Given $f \in C^1[a, b]$, find $s \in PP_3(\pi)$ so that

1. $s(x_i) = f(x_i)$, $x_i \in$[a,b] , $0 \leq i \leq n$;
2. $s'(x_o) = f'(x_o)$ and $s'(x_n) = f'(x_n)$;
3. s is twice continuously differentiable,

It can be shown that there is a unique $s \in \mathrm{S}_3(\pi)$ satifying all conditions above, and that the dimension of $\mathrm{S}_3(\pi)$ is $n + 3$.

3 $\mathrm{H}_3(\pi)$ space

Let's first define this space as follows :

Consider $f \in C^1[a,b]$ and π a partition of $[a,b]$. There is a unique function $s \in PP_3(\pi)$ satisfying the following interpolation problem :

$$f^{(j)}(x_i) = s^{(j)}(x_i), \qquad 0 \le i \le n \quad \text{and} \quad j = 0,1 \tag{3}$$

Functions $s \in PP_3(\pi)$ satisfying this property are called cubic Hermite functions, and the space spanned by them is denoted by $\mathrm{H}_3(\pi)$. It can be shown that the dimension of $\mathrm{H}_3(\pi)$ is $2(n+1)$.

In the following we will summarize the local basis to $\mathrm{H}_3(\pi)$ space (see [1]). A local basis covers a single arbitrary subinterval (or *element*), while the global basis covers all the partition. The use of a local basis instead a global one, associated with master and real elements is typical of finite element computations; that idea will be helpful in this work.

3.1 Local Basis in the Master Element

When we deal with functions if one real variable, the *master element* is the interval [-1,1], that we will denote by $\hat{\mathrm{K}}$.

$$s(t)\mid_{\hat{\mathrm{K}}} = \sum_{i=1}^{4} \alpha_i \hat{\Phi}_i(t), \qquad t \in [-1,1] \ , \tag{4}$$

$$\hat{\Phi}_1(t) = (2+t)(1-t)^2 * 0.25 \tag{5}$$
$$\hat{\Phi}_2(t) = (1+t)(1-t)^2 * 0.25 \tag{6}$$
$$\hat{\Phi}_3(t) = (2-t)(1+t)^2 * 0.25 \tag{7}$$
$$\hat{\Phi}_4(t) = (t-1)(1+t)^2 * 0.25 \tag{8}$$

This basis will be helpful for computation tasks.

3.2 Local Basis in the Real Element

The *real element* is a generic interval $[x_k, x_{k+1}]$ of the partition π.

$$s(x)\mid_{K} = \sum_{i=1}^{4} \alpha_i \Phi_{i(k)}(x), \qquad x \in [x_k, x_{k+1}] \tag{9}$$

$$\Phi_{1(k)}(x) = \frac{(x_{k+1}-x)^2}{(x_{k+1}-x_k)^3}[2(x-x_k)+(x_{k+1}-x_k)] \tag{10}$$

$$\Phi_{3(k)}(x) = \frac{(x-x_k)^2}{(x_{k+1}-x_k)^3}[2(x_{k+1}-x)+(x_{k+1}-x_k)] \tag{11}$$

$$\Phi_{2(k)}(x) = \frac{(x-x_{k+1})^2(x-x_k)}{(x_{k+1}-x_k)^2} \tag{12}$$

$$\Phi_{4(k)}(x) = \frac{(x-x_k)^2(x-x_{k+1})}{(x_{k+1}-x_k)^2} \tag{13}$$

3.3 Master and Real Elements Connection

We define the transformation

F: [-1,1] $\longrightarrow [x_k, x_{k+1}]$

$t \longmapsto \mathrm{F}(t) = at + b$, satisfying $\begin{cases} \mathrm{F}(-1) = x_k \\ \mathrm{F}(1) = x_{k+1} \end{cases}$

We have $a = \frac{x_{k+1}-x_k}{2}$ and $b = \frac{x_k+x_{k+1}}{2}$.

F associates to each point t of [-1,1] another point x in $[x_k, x_{k+1}]$. We can now associate the functions $\hat{\Phi}_i$ with those defined in $[x_k, x_{k+1}]$. Setting x = F(t):

$$\begin{cases} \Phi_i(x) = \hat{\Phi}_i(t) \;\;,\;\; t \in [-1,1] \;\;,\;\; i = 1,3 \\ \\ \Phi_i(x) = \hat{\Phi}_i(t) \cdot a \;\;,\;\; t \in [-1,1] \;\;,\;\; i = 2,4 \end{cases} \tag{14}$$

Derivating twice with respect to t, we have :

$$\begin{cases} \frac{\mathrm{d}^2}{\mathrm{dx}^2}\Phi_i(x) = \frac{\mathrm{d}^2}{\mathrm{dt}^2}\hat{\Phi}_i(t)\frac{1}{a^2} \;\;,\;\; i = 1,3 \\ \\ \frac{\mathrm{d}^2}{\mathrm{dx}^2}\Phi_i(x) = \frac{\mathrm{d}^2}{\mathrm{dt}^2}\hat{\Phi}_i(t)\frac{1}{a} \;\;,\;\; i = 2,4 \end{cases} \tag{15}$$

4 Relation between $\mathrm{H}_3(\pi)$ and $\mathrm{S}_3(\pi)$

We can define $\mathrm{H}_3(\pi)$ in a generic way. By the last definition, all functions belonging to this space are necessarily C^1 , since on the possible descontinuity points – the nodes of π – we have a continuity restriction (due to interpolatory conditions).
It sugests another way to define $\mathrm{H}_3(\pi)$:

$$\mathrm{H}_3(\pi) = PP_3(\pi) \cap C^1[a,b] \tag{16}$$

Since every C^2 function is also a C^1 function ($C^2[a,b] \subset C^1[a,b]$), we expect that every cubic spline is also a cubic Hermite. The following theorem, that uses standard arguments from interpolation proofs, stabilishes that $\mathrm{S}_3(\pi) \subset \mathrm{H}_3(\pi)$.

Theorem 1. For all $f \in \mathrm{S}_3(\pi)$ there is a unique $s \in \mathrm{H}_3(\pi)$ so that $f = s$.

Proof. Let $f \in \mathrm{S}_3(\pi)$ and $s \in \mathrm{H}_3(\pi)$ be so that

$$s^{(j)}(x_i) = f^{(j)}(x_i), \qquad 0 \le i \le n \text{ and } j = 0,1 \text{ } (s \text{ is unique}) \tag{17}$$

Defining $g = f - s$, we have:

- $g \in PP_3(\pi)$, since $f, s \in PP_3(\pi)$
- $g,\ g'$ are continous, since f and $s \in C^1[a,b]$
- $g(x_i) = g'(x_i) = 0$, $0 \le i \le n$

By Rolle Theorem,

$$g'(\bar{x}_k) = 0, x_k \le \bar{x}_k \le x_{k+1}, 0 \le k \le n-1 \tag{18}$$

Therefore, points $x_0, \bar{x}_0, x_1, \ldots, \bar{x}_{n-1}, x_n$ are roots of g' , so in each subinterval $[x_k, x_{k+1}]$, $g' \mid_K$ have three roots: x_k, $\bar{x}_k$ and x_{k+1}, $0 \le k \le n-1$.

As $g' \mid_K$ is a polynomial with degree two, by the Algebraic Fundamental Theorem, $g' \mid_K \equiv 0$, $0 \le k \le n-1$.

As $g' \mid_K \equiv 0$, $g \mid_K$ is a polynomial with degree one. But in each subinterval $[x_k, x_{k+1}]$ the function g have two roots: x_k and x_{k+1} , and by Algebraic Fundamental Theorem, $g \mid_K \equiv 0$, $0 \le k \le n-1$.

Therefore, $g \equiv 0$ and $f = s$.

5 Deriving the Cubic Splines

We will find the splines selecting a subset of $\mathrm{H}_3(\pi)$, according to the last result. The selecting criterion will be the continuity of the second derivative at each interior point x_k of the partition π:

$$\frac{\mathrm{d}^2}{\mathrm{dx}^2} s_-(x_k) = \frac{\mathrm{d}^2}{\mathrm{dx}^2} s_+(x_k) \quad , \quad 1 \leq k \leq n-1 \quad , \tag{19}$$

where $s_-(x) = s(x) \mid_{K-1}$ and $s_+(x) = s(x) \mid_K$.
Setting with the local basis

$$S(x) \mid_K = y_k \Phi_{1(k)}(x) + y_{k+1} \Phi_{3(k)}(x) + \sigma_k \Phi_{2(k)}(x) + \sigma_{k+1} \Phi_{4(k)}(x) \quad , \tag{20}$$

we have from (19):

$$\begin{aligned} y_{k-1} \frac{\mathrm{d}^2}{\mathrm{dx}^2} \Phi_{1(k-1)}(x_k) + y_k \frac{\mathrm{d}^2}{\mathrm{dx}^2} \Phi_{3(k-1)}(x_k) + \sigma_{k-1} \frac{\mathrm{d}^2}{\mathrm{dx}^2} \Phi_{2(k-1)}(x_k) + \\ + \sigma_k \frac{\mathrm{d}^2}{\mathrm{dx}^2} \Phi_{4(k-1)}(x_k) \;=\; y_k \frac{\mathrm{d}^2}{\mathrm{dx}^2} \Phi_{1(k)}(x_k) + y_{k+1} \frac{\mathrm{d}^2}{\mathrm{dx}^2} \Phi_{3(k)}(x_k) + \\ + \sigma_k \frac{\mathrm{d}^2}{\mathrm{dx}^2} \Phi_{2(k)}(x_k) + \sigma_{k+1} \frac{\mathrm{d}^2}{\mathrm{dx}^2} \Phi_{4(k)}(x_k) \end{aligned} \tag{21}$$

Instead operating with the functions Φ_i, we will do all computations on master element, and then use the transformation defined on section 3.3 .
Procceding this way,

$$\begin{array}{ll} \frac{\mathrm{d}^2}{\mathrm{dx}^2} \hat{\Phi}_1(t) = \frac{3}{2} t & \frac{\mathrm{d}^2}{\mathrm{dx}^2} \hat{\Phi}_2(t) = \frac{3t-1}{2} \\ \frac{\mathrm{d}^2}{\mathrm{dx}^2} \hat{\Phi}_3(t) = \frac{-3}{2} t & \frac{\mathrm{d}^2}{\mathrm{dx}^2} \hat{\Phi}_4(t) = \frac{3t+1}{2} \end{array} \tag{22}$$

Defining $h_k = x_{k+1} - x_k$, we can write (15) as:

$$\begin{cases} \frac{\mathrm{d}^2}{\mathrm{dx}^2} \Phi_{i(k)}(x) = \frac{\mathrm{d}^2}{\mathrm{dx}^2} \hat{\Phi}_i(t) \frac{4}{h_k^2} \quad , \quad i = 1, 3 \\ \frac{\mathrm{d}^2}{\mathrm{dx}^2} \Phi_{i(k)}(x) = \frac{\mathrm{d}^2}{\mathrm{dx}^2} \hat{\Phi}_i(t) \frac{2}{h} \quad , \quad i = 2, 4 \end{cases} \tag{23}$$

It follows that:

$$\begin{array}{ll} \frac{\mathrm{d}^2}{\mathrm{dx}^2} \Phi_{1(k)}(F(t)) = \frac{6}{h_k^2} t & \frac{\mathrm{d}^2}{\mathrm{dx}^2} \Phi_{2(k)}(F(t)) = \frac{3t-1}{h_k} \\ \frac{\mathrm{d}^2}{\mathrm{dx}^2} \Phi_{3(k)}(F(t)) = \frac{-6}{h_k^2} t & \frac{\mathrm{d}^2}{\mathrm{dx}^2} \Phi_{4(k)}(F(t)) = \frac{3t+1}{h_k} \end{array} \tag{24}$$

As F(-1) = x_k and F(1) = x_{k+1} :

$$\begin{array}{lcl} \frac{d^2}{dx^2}\Phi_{1(k)}(x_k) = \frac{-6}{h_k^2} & \text{and} & \frac{d^2}{dx^2}\Phi_{1(k)}(x_{k+1}) = \frac{6}{h_k^2} \\ \frac{d^2}{dx^2}\Phi_{2(k)}(x_k) = \frac{-4}{h_k} & \text{and} & \frac{d^2}{dx^2}\Phi_{2(k)}(x_{k+1}) = \frac{2}{h_k} \\ \frac{d^2}{dx^2}\Phi_{3(k)}(x_k) = \frac{6}{h_k^2} & \text{and} & \frac{d^2}{dx^2}\Phi_{3(k)}(x_{k+1}) = \frac{-6}{h_k^2} \\ \frac{d^2}{dx^2}\Phi_{4(k)}(x_k) = \frac{-2}{h_k} & \text{and} & \frac{d^2}{dx^2}\Phi_{4(k)}(x_{k+1}) = \frac{4}{h_k} \end{array} \tag{25}$$

The procedure for $[x_{k-1}, x_k]$ is the same, but h_{k-1} replaces h_k.
Updating (21), for $1 \leq k \leq n-1$, we have:

$$\frac{6y_{k-1}}{h_{k-1}^2} - \frac{6y_k}{h_{k-1}^2} + \frac{2\sigma_{k-1}}{h_{k-1}} + \frac{4\sigma_k}{h_{k-1}} = \frac{-6y_k}{h_k^2} + \frac{6y_{k+1}}{h_k^2} - \frac{4\sigma_k}{h_k} - \frac{2\sigma_{k+1}}{h_k} \tag{26}$$

These restrictions will cause loss of freedom degrees, that is, some unknowns will be combination of others, so that $S_3(\pi)$ will have a dimension lower than $H_3(\pi)$, as we have expected.

The coefficients y_i and σ_i, $0 \leq i \leq n$, are related to the interpolation of the function and the derivative, respectively (according to the definition of $H_3(\pi)$). As most of the problems involves interpolation of the function, it is convenient to find the unknowns σ_i as combinations of y_i.

Rearranging (26) :

$$h_k\sigma_{k-1} + 2(h_k + h_{k-1})\sigma_k + h_{k-1}\sigma_{k+1} = f(y_k) \ \ , \ \ 1 \leq k \leq n-1 \ \ , \tag{27}$$

where $f(y_k)$ is given by:

$$f(y_k) = 3\left[\frac{-h_k}{h_{k-1}}y_{k-1} + \left(\frac{h_k}{h_{k-1}} - \frac{h_{k-1}}{h_k}\right)y_k + \frac{h_{k-1}}{h_k}y_{k+1}\right] \tag{28}$$

We have n-1 equations and n+1 unknowns. There are two frequent ways to supply the remaining equations: extending the partition π with two artificial nodes, one in the right, other in the left, or restrict the derivative on the extreme nodes, x_0 and x_n (allowing interpolant conditions at $s'(x_0)$ and $s'(x_n)$). We use the last one:

$$\begin{array}{lcl} s'(x_0) = \bar{y}_0 & \rightarrow & \sigma_0 = \bar{y}_0 \\ s'(x_n) = \bar{y}_n & \rightarrow & \sigma_n = \bar{y}_n \end{array} \tag{29}$$

The equalities in (29) suggest us to eliminate the unknowns σ_0 and σ_n from the system. The reduced system in the matrix form $A.x = b$ is:

$$A = \begin{bmatrix} 2(h_1+h_0) & h_0 & & & \mathbf{0} \\ h_2 & 2(h_2+h_1) & h_1 & & \\ & \ddots & \ddots & & \\ & h_{n-2} & 2(h_{n-2}+h_{n-3}) & h_{n-3} & \\ \mathbf{0} & & h_{n-1} & 2(h_{n-1}+h_{n-2}) \end{bmatrix} \quad (30)$$

$$x = (\sigma_1 \ , \ \sigma_2 \ , \ \ldots \ , \ \sigma_{n-2} \ , \ \sigma_{n-1})^T \quad (31)$$

$$y = (f(y_1) - h_1\bar{y}_n \ , \ f(y_2) \ , \ \ldots \ , \ f(y_{n-2}) \ , \ f(y_{n-1}) - h_{n-2}\bar{y}_n)^T \quad (32)$$

Observe that

$$|\, a_{i,i} \,| = |\, 2(h_i + h_{i-1}) \,| > h_i + h_{i-1} \geq |\, a_{i,i-1} \,| + |\, a_{i,i+1} \,| \quad (33)$$

As $a_{i,j} = 0, |\, i - j \,| > 1$,

$$|\, a_{i,i} \,| > \sum_{j \neq i} |\, a_{i,j} \,| \quad , \quad (34)$$

that is, the matrix A is strictly diagonally dominant by column, so it is non singular. Therefore the system above have a unique solutuion, and σ_i are completely determined by $\bar{y}_0$, $y_i \ , \ 0 \leq i \leq n$ e $\bar{y}_n$.

6 Conclusion

In the following we present an algorithm that uses the method above, written for the software $MATLAB^{©}$. Two subroutines (**solvetrd.m** and **basis.m**) join the main function (**h_spline.m**). There is also an example of spline interpolation using these routines. Therefore the derivation of splines presented not only serves as a learning alternative but also provides another computer method for applications envolving splines.

function h_spline(x,y)

```
% h_spline(x,y) :  spline interpolation using hermite functions
% x :  nodes of the partition
% y :  interpolation table; the 1st and the last ones interpolate
% the derivative at the extremal points

% Setting parameters

  n = length(x) - 1;
  if(length(y) =n+3)
    error('Wrong input arguments:  x and y are not compatible');
  end

  m = 16;          % spacing parameter for the plot

% The tridiagonal linear system ( See equations (30) to (32) )

  % Computing h

  for k = 1:n
    h(k) = x(k+1) - x(k);
  end

  % Computing the rigth hand size ( See equations (28) and (32) )

  b(1) = y(1);

  for k = 1:n-1
    aux = h(k+1)/h(k);
    aux2 = 1/aux;
    b(k) = 3*( -aux*y(k+1) + (aux-aux2)*y(k+2) + aux2*y(k+3) );
  end

  b(1) = b(1) - h(2)*y(1);
```

```
  b(n-1) = b(n-1) - h(n-1)*y(n+3);

  % Computing the diagonals of the system matrix

  for i=1:n-2
    Sub(i) = h(i+2);
    Diag(i+1) = 2*( h(i+1) + h(i) );
    Sup(i) = h(i);
  end

  Diag(n-1) = 2*( h(n) + h(n-1) );

  % Computing the variables sigma

  sigma(1) = y(1);
  sigma(2:n) = solvetrd(Sub,Diag,Sup,b);

  sigma(n+1) = y(n+3)
% Assembling and drawing the solution

  % Definig the domain in the variable t

  j = 1;

  for k = 1:n
    t(j) = x(k);
    j = j + 1;
    Spacing = h(k)/m;
    for i = 1:m-1
      t(j) = t(j-1) + Spacing;
      j = j + 1;
    end
  end

  t(j) = x(n+1);    % the last point of the domain
  n_t = j;          % dimension of t
```

```
  % assembling the solution with the hermite local basis
  % see equation (20)

  j = 1;

  for k = 1:n
    for i = 1:m
      phi = basis(t(j),k,x,h);
      aux = y(k+1)*phi(1) + y(k+2)*phi(3);
      spl(j) = aux + sigma(k)*phi(2) + sigma(k+1)*phi(4);
      j = j + 1;
    end
  end

  spl(n_t) = y(n+2);

  % drawing

  plot(t,spl,x,y(2:n+2),'o');
```

function [phi] = basis(xo,k,x,h)

```
% basis :  hermite functions local basis evaluated at x = xo
% See equations (10) to (13)
% The solution is returned in the vector phi
% xo :  point of evaluation
% k  :  subinterval index
% x  :  partition vector
% h  :  h(k) = x(k+1) - x(k)

  aux = x(k+1) - xo;
  aux2 = xo - x(k);

  phi(1) = aux^2*(2*aux2 + h(k))/h(k)^3;
```

```
  phi(2) = aux^2*aux2/h(k)^2;
  phi(3) = aux2^2*(2*aux + h(k))/h(k)^3;
  phi(4) = -aux*aux2^2/h(k)^2;
```

function x = solvetrd(Sub,Diag,Sup,b);

```
% solvetrd :  tridiagonal system solver
% The solution is returned in x
% Sub  :  lower diagonal
% Diag :  main diagonal
% Sub  :  upper diagonal
% b    :  right hand side

  N = length(b);

% forward elimination

  for i = 2:N
    Diag(i) = Diag(i) - Sup(i-1)*Sub(i-1)/Diag(i-1);
    b(i) = b(i) - b(i-1)*Sub(i-1)/Diag(i-1);
  end

% back substitution

  x(N) = b(N)/Diag(N);

  for i = 1:N-1
    x(N-i) = (b(N-i) - Sup(N-i)*x(N-i+1))/Diag(N-i);
  end
```

7 References

[1] P. M. Prenter - *Splines and Variational Methods*, 1975, John Wiley & Sons.

[2] George E. Forsythe - *Computer Methods for Mathematical Computations*, 1977, Prentice-Hall.

[3] Larry L. Schumacker - *Spline Funcions: Basic Theory*, 1981, John Wiley & Sons.