Table of Contents.

Intro 2

Mathematics of "spline function" calculation 3

Change of parametrization of the "spline function" 4

Case of the "spline function" with defined edge slopes 5

Study of the case where an interval is of zero length, first approach 6

Question, is the spline dependent on the choice of reference frame? 7

A first summary: 10

Case G1 where the existence of the second derivative is not desired. 11

Case G1 and with discontinuity of a tangent. 12

Another way to approach the calculation of the spline function, *M i* = second derivative 13

Let us study the situation when the slopes at the edges are fixed. 15

In the case where the slopes are given at the edges, here is the matrix to solve 16

Study of the case where an interval has zero length. 17

Take the case where we do not want continuity of the derivatives at *t* = *t* 4  18

Case of periodic spline, corresponding to closed curves. 19

Case of periodic spline, corresponding to closed curves, but with break. 20

Links with other spline curves, described in the video “The continuity of Splines. » 21

More precise definition of a B-spline and links with Math-splines 22

Determination of the two control points *V* 0 and *V n* +1 of the Math-spline   
to make the two curves coincide. 25

Writing in matrix form of the transition from *U i* to *V i* 27

Closed B-splines 29

Study of the general case, where the transit times *t i* are no longer regular 30

Lots of calculations...

**Summary - result - conclusion of previous calculations 41**

How to determine the two control points *V* 0 and *V* n+1 of the Math-spline   
to make the two curves coincide? 42

Writing in matrix form of the transition from *U i* to *V i* 44

Verifications! 45

Closed B-splines in the case of non-regular times *t i* 49

Table of different cases of comparisons of B-splines and Math-splines 50

Appendix I, approximation of the second derivative of a function from 3 points. 51

Annex II. Verification that the curve obtained by B-spline is   
twice continuously differentiable 52

Introduction

A few decades ago I learned about degree 3 spline functions and wrote programs that use them to pass a curve through given points in a plane. I've been waiting for such curves to be implemented in software like Inkscape and FreeCAD for a long time.

Freya Holmér   
's excellent video on Youtube called: “The continuity of Splines. »

See: <https://www.youtube.com/watch?v=jvPPXbo87ds>

by Freya Holmér see: <https://www.youtube.com/@Acegikmo>

Unfortunately, the curve I would name "Math-spline", based on the spline functions that I describe later and that I implemented in the following web page:

<https://www.juggling.ch/gisin/bgweb/aprod2000_perso/spline_curve_math.html>  
is not described by Freya Homér nor used in the software I know.

Here are some characteristics of this “Math-spline” curve:

° continuous

° of continuously varying tangents along the curve

° of radii of curvature varying continuously along the curve, it is G 2

° the influence of the points is practically on the 8 neighboring segments of the point

° is easy to calculate, quickly

° can easily be closed

° can have breakpoints, so where the tangent varies discontinuously.

° is invariant under rotation, symmetry and dilation

We will suppose to have a list of *n* = nbPts points in the plane, for *i* = 1.. nbPts

We want to pass a “natural” curve through these points. To do this, we will define the intervals   
= distance between and = and the times , with and we will determine two "spline functions" determined by the data: ( *t i* ; *x i* ) and ( *t i* ; *y i* ), for *i* = 1. .nbPts.

We will see two ways of calculating these “spline functions”, how to make them periodic, how to introduce breaks in the curve and that they are invariant under rotation, symmetry and dilation.

Mathematics of "spline function" calculation

Reference : <http://www.unige.ch/~hairer/polycop.html>

" Handouts of the course "Numerical Analysis" (June 2005), Chapter II. Interpolation and Approximation » pages 46 to 52. ( <http://www.unige.ch/~hairer/poly/chap2.pdf>)

This is the first way I learned, but there is another one that I prefer. It is described further on page 11 and is the one I use in the program.

Data :

( *t i* ; *y i* ), for *i* = 1.. nbPts

Spline passing through the points, cubic by piece, twice continuously differentiable:

, for *i* = 1.. nbPts

, for *i* = 1.. nbPts - 1

For *i* = 1.. nbPts - 1,

Where *p* i is the slope of the spline *at t* = *t i* .

So .

So and .

Verifications:

;

, with negligible, so *p i* is indeed the slope at *t* = *t i* .

, with negligible, so *p i+1* is indeed the slope at *t* = *t i+1* .

Condition to be met for the second derivative to be continuous:

and

It is necessary that :

So :

Multiply by and rearrange the terms:

Condition to be fulfilled for *i* = 2.. nbPts - 1.

For “Natural Splines”, we have: , so

, or also : and

, or also : and are arbitrary nonzero.

System of equations to solve to obtain the value of *p i* *i* = *1..n*

*n* = number of points = nbPts

*n* = 7 in the matrix example above which is an *n* x *n matrix*

; , for *i* = 1.. nbPts - 1

*i* =2..n *-* 1

and .

We have used the case of the “natural” spline, which is defined by the characteristic that its second derivative at the edges is zero.

Resolution :

; ( and would be natural)

; ;

If , then *i* =2..n *-* 1

*i =* 0..n

for i=2 to *n* do and we have:

for i = *n* -1 downto 1 do we have:

Change of parameterization of the spline function.

For *i* = 1.. nbPts - 1,

Case of the "spline function" having the slopes at the edges defined

We assume given the slopes at the edges: = given; = given,

*n* = 8 in the matrix example above

*i* =3..n *-* 2

and

For a **periodic spline** , the easiest way is to increase the matrix by 10 points, which merge with the points to give a closed curve which intersects, then calculate the *pi , then* eliminate the first 5 and the last 5.

The case of the periodic spline is taken up later, after having seen another approach to calculating the spline, which will make the task easier.

Study of the case where an interval is of zero length, first approach.

Further after determining another way to calculate the "spline function", the case where the interval is of zero length is restated and is simpler. So the following can be skipped.

Take the case where , in this case the matrix becomes:

and

In this case, lines 4 and 5 come down to a system of two equations with two unknowns:

and

Workaround: .

If , the value of is undetermined.

By setting it equal to 0 ( ), we have the advantage that the segment going from *t 4* to *t 5* remains constant and that a break can occur in this segment of zero and constant length, merged with the points .

On the other hand, since the values of and are known, the matrix breaks down into two independent matrices. It is practically useless, but it had to be noticed.

Let's remember that :

So = constant

= 0 =>

= 0 =>

Lines 4 and 5 can be replaced by the lines above, to modify the system and have natural boundary conditions. This will be easier with the other way of determining the “spline function”.

Question, is the spline dependent on the choice of reference frame?

In other words,   
if we apply a linear transformation to the points defining the spline, then we determine the spline from these points or   
if we determine the spline from the points, then we apply the same linear transformation to it,   
obtains- do we have the same curve?

The following shows that yes if the transformation is a rotation, a symmetry or a dilation.

This was also checked programmatically.

Note: the 2x2 linear transformation matrix.

It will be necessary to imagine two cases, one where the transformation is orthogonal, that is to say that it preserves the distances and the general case.

Note: the *n x 2* matrix of points defining the spline.

Note: = the ( *n -1) x 2* matrix defining the curve of the spline.

is a function *F* of *t* and of . It will be developed further.

Note: = the ( *n -1) x 2* matrix defining the curve of the spline after its linear transformation.

Note: the *n x 2* matrix of points defining the spline, after linear transformation.

The question is to know if is obtained by the same function *F* of *t* and of ?

Let us explain the function *F .*

*= + + +*

and . is the vector obtained from the TL of the points.

We saw on the previous page that: , because

== \_

= where is the large *n* x *n* matrix from page 2.

= where

*n* x *n* matrix defining the values of the *q i* which are defined on page 2.

It only depends on the values of *h i* , which are either independent of the points, or only dependent on the distances between the points.

Result :

== \_

and if is the vector obtained from the TL of the points, then

=

= = .

So we have the relation : = !!!

We can do the same to show that:

if and are the vectors obtained from the TLs of the points, then

= and = !!!

This allows us to obtain:

The linear transformation of the spline:

= + + +

+ + + =   
is equal to the spline obtained from the points which have undergone the linear transformation.

This shows that if the matrices ; ; ; ...   
are independent of the points, then the curve of the spline is independent of the choice of reference frame, because to make the points undergo a linear transformation then calculate the spline or calculate the spline then make it undergo the linear transformation is the same thing.

In the case where we choose *h i* = 1 for all *i* , we do have independence at the points.

In the case where one chooses *h i* =distance between the point *p i* and the point *p i* +1, the independence is true only if the linear transformation is orthogonal, for example a rotation or a symmetry.

A first summary:

The spline curve defined by the previous calculations is:

° continuous

° of continuously varying tangents along the curve

° of radius of curvature varying continuously along the curve

° the influence of the points is practically on the 8 neighboring segments of the point

° is easy to calculate, quickly

° can easily be closed

° can have breakpoints, so where the tangent varies discontinuously.

Some justifications for the above statements.

Between two points, the coordinates of the points of the spline are given by a cubic polynomial, so between two points the curve is perfectly smooth, even from the mathematical point of view.

It remains to see that the statements are correct at the points of definition of the spline.

The continuity is obvious, because the spline passes through the given points.

The way to construct the spline curve involves the construction of two functions that are twice continuously differentiable, sX(t) and sY(t), which define the parametric equation of the spline.

(X; Y) = (sX( *t* ); sY( *t* )) for *t* varying from 0 to *t nbPts* .

Slope of the tangent at *t* = , which varies continuously.

When the denominator is zero, the slope is simply vertical.

The radius of curvature at *t* = , which varies continuously.

When the denominator is zero, the radius of curvature is infinite.

It varies continuously, because the two coordinates are twice continuously differentiable.

Case G 1 where the existence of the second derivative is not desired.

To simplify, we may not want the curve to be continuously differentiable twice, but just continuously differentiable once. We just have to choose reasonable values of *p i* .

We want *p i* to approximate “at best” the derivative of the curve at *t i* .

*t*0

*tp*

*tm*

Let *f be* a C 2 function , hence twice continuously differentiable on the interval [ *t m* ; *p* ] *\_*

*t m* < *t* 0 < *t p* given.

With: and , both positive.

Knowing ; and , we would like to approximate " at best " .

We have :

where

where

We can think that and are close, we don't have better, so a choice that seems reasonable to me is: and .

So we approximate by:

A less good, but simpler choice is: and . which   
approximates by .

Case G 1 and with discontinuity of a tangent.

*t*3

*s*3

*s*4

*t*5

*t*6

*t*2

*t*4

*t*1

If there is a break at *t* 4 , *s* 3 only depends on the points at *t* 2 , *t* 3 , and *t* 4 , but not at *t* 5 .

If there is a break at *t* 4 , *s* 4 only depends on the points at *t* 4 , *t* 5 , and *t* 6 , but not at *t* 3 .

(If there is a break at *t* 3 , *s* 3 only depends on the points at *t* 3 , *t* 4 , and *t* 5 , but not at *t* 2 .)

In the case of a break in *t* 4 .

We would like to have

, So :

Here we refer to *p* 4 of *s* 3 , which is different from *p* 4 of *s* 4 .

We would like to have

, So :

Here we refer to *p* 4 of *s* 4 , which is different from *p* 4 of *s* 3 .

In the case of a break at *t* 3 .

We would like to have

, So :

Here we refer to the *p* 3 of *s* 3 , which is different from the *p* 3 of *s* 2 .

Another way of approaching the calculation of the spline function, *M i* = second derivative.

Reference: “Introduction to numerical analysis” by J. Stoer and R. Bulirsch, Springer-Verlag 1983.

ISBN: 0-387-90420-4 New York or 3-540-90420-4 Berlin Heidelberg.

This way has advantages over the one seen in the previous pages.

Data :

( *t i* ; *y i* ), for *i* = 1.. nbPts

Spline passing through the points, cubic by piece, twice continuously differentiable:

, for *i* = 1.. nbPts

, for *i* = 1.. nbPts - 1

For *i* = 1.. nbPts - 1,

Where M *i* is the second derivative of the spline *at t* = *t i* .

So and .

So the second derivative of s(t) is continuous, provided that the derivative is continuous.

At the edges:

Condition for the derivative to be continuous: , therefore:

What becomes by rearranging the terms:

for *i* = 2.. nbPts - 1

Change of parameterization of the curve.

For *i* = 1.. nbPts - 1,

before: , so:

before: , so:

before :

In matrix form, this becomes

System of equations to be solved to obtain the value of *M i* *i* =2..n *-* 1

*n* = number of points = nbPts

*M1* = *Mn* = *0* \_

These two equalities correspond to the case of the "natural" spline, which is defined by the characteristic that its second derivative at the edges is zero.

*n* = 9 in the matrix example above

, for *i* = 1.. nbPts - 1

= 0 if the numerator is zero, even if the denominator is also zero. It's arbitrary, but practical.

*i* =2..n *-* 1

Resolution :

; If , then *i* = 2.. *n* -1 The first line is *i* =2

*i* = 1.. *n* -1

for *i* =3 to *n* -1 do and we have:

for *i* = *n* -2 downto 2 do we have:

What if in the algorithm, we divide by 0?

This would only be the case if and , so three points are superimposed.

In this case, the whole row of the matrix would be zero, which would have no influence, because the length of the segment to be drawn would also be zero. This problem is avoided by making the zero diagonals equal to 1.

Let us study the situation when the slopes at the edges are fixed.

We assume given *p* 1 and *p n* . This will influence the values of *M* 1 and *M n* .

We have seen that for *i* = 1.. nbPts - 1

and .

So =>

We must also satisfy:

Combining, we get: .

This changes the value of the first number on the diagonal, as well as the value of *r* 2 .

If we have chosen the *“natural” situation where M 1 = 0* , then: ( *y* 0 = *y* 1 - *p* 1 )

In any case, we have: ( *V* 0 = *V* 1 - *p* 1 )

At the end of the curve:

We have seen that:

So =>

We must also satisfy:

Combining, we get: .

This changes the value of the last number on the diagonal, as well as the value of *r n* -1 .

If we have chosen the *“natural” situation where M n = 0* , then: ( *y* n+1 = *y n* + *p* n )

In any case, we have: ( *V* n+1 = *V n* + *p* n )

Another possibility is to increase the dimension of the matrix to be solved by two.

In the case where the slopes are given at the edges, here is the matrix to solve:

*n* = 7 in the matrix example above

, for *i* = 1.. nbPts - 1

= 0 if the numerator is zero, even if the denominator is also zero. It's arbitrary, but practical.

*i* =2..n *-* 1

;

Resolution :

;

; If , then *i* = 1.. *n* The first line is *i* =1

*i* = *1..n*

for *i* =2 to *n* do and we have:

for *i* = *n* -1 downto 1 do we have:

There are *n* + 2 points; *n* = nb\_points - 2.

Study of the case where an interval has zero length.

Take the case where *h* 4 = 0, in this case the matrix becomes:

*n* = 9 in the matrix example above

, for *i* = 1.. nbPts - 1

= 0 if the numerator is zero, even if the denominator is also zero. It's arbitrary, but practical.

*i* =2..n *-* 1

The system splits into two independent systems.

is indeterminate.

So *r* 4 and *r* 5 are also indeterminate.

It would be practical and natural to have *M* 4 = *M* 5 = 0, so that the spline breaks down into two natural splines.

We would have :

. The equation: would be replaced by *M* 4 = 0.

. The equation: would be replaced by *M* 5 = 0.

Resolution :

; If , then *i* = 2.. *n* -1 The first line is *i* =2

*i* = 1.. *n* -1

and add to the algorithm.

for *i* =3 to *n* -1 do and we have:

for *i* = *n* -2 downto 2 do we have:

We automatically have the two desired equalities: *M* 4 = *M* 5 = 0.

Take the case where we do not want continuity of the derivatives at *t* = *t* 4

This would make it possible to have a break in the slope at *t* 4 .

The strikethrough line in the system below should be eliminated, leaving a degree of freedom.

*M* 4 of segment *s* 4 need not be equal to *M* 4 of segment *s* 5 .

It would be natural to set *M* 4 = 0, so that the spline breaks down into two natural splines.

*n* = 9 in the matrix example above

, for *i* = 1.. nbPts - 1

= 0 if the numerator is zero, even if the denominator is also zero. It's arbitrary, but practical.

*i* =2..n *-* 1

Resolution :

We leave the line crossed out, *i* =4, which will automatically give: *M* 4 = 0.

; If , then *i* = 2.. *n* -1 The first line is *i* =2

*i* = 1.. *n* -1

and add to the algorithm.

for *i* =3 to *n* -1 do and we have:

for *i* = *n* -2 downto 2 do we have:

We automatically have the desired equality : *M* 4 = 0.

Case of periodic spline, corresponding to closed curves.

In this case, the second derivative of the curve at the “starting point” in *M* 1 and   
the second derivative of the curve at the “end point” in *M* n+10 are approximated .

We add 10 points periodically, we eliminate the first 5 and last 5 segments, which gives us the periodic curve, because beyond 5 points, the segment is no longer influenced by the points.

Approximation of the second derivatives at the start and at the finish:

For a curve passing through points it is different than for a spline function.

Here we deal with the case of a curve passing through points.

Given: for *i* = 1.. nbPts

We add: and we can add: .

= distance between and = = slope , for *i* = 0.. nbPts +10

and *n* = nbPts and *m* = *n* + 10

System of equations to be solved to obtain the value of *M i* *i* = *2..n* +10-1

*n* = number of points = nbPts

*m* = 8 in the matrix example above

*i* = 3.. *m* -2

and

Resolution :

; If , then *i* = 2.. *m* -1 The first line is *i* =2

*i* = 1.. *m* -1

for *i* =3 to *m* -1 do and we have:

for *i* = *m* -2 downto 2 do we have:

We only keep the values from to . Note   
to define segments from 1 to *n* .

Case of periodic spline, corresponding to closed curves, but with break.

In the case of a closed curve, with a break at a point, that is to say with a point having a discontinuity of the tangent, this corresponds to a non-periodic spline, with a beginning and an end superimposed on the point having the break.

For the calculations, you just have to start the matrix at the point having the break.

The other way is to add the following conditions to each breakout *k* :

and to add in the algorithm, before the elimination of Gauss and to add points, as described previously.

It's easier.

Links with other spline curves, described in the video “The continuity of Splines. »

See: <https://www.youtube.com/watch?v=jvPPXbo87ds>

by Freya Holmér see: <https://www.youtube.com/@Acegikmo>

Excellent video which summarizes different approximations by splines.

Unfortunately, the spline I described earlier is not part of the list of splines described in this video.

Bezier:

P1 et P2 sont des points de contrôle.

P(0) = P0

P(1) = P3

P’(0) = -3 P0 + 3 P1

P’(1) = -3 P2 + 3 P3

Hermit:

P(0) = P0

P(1) = P1

P’(0) = V0

P’(1) = V1

Catmull-Rom:

P(0) = P1

P(1) = P2

P’(0) = -0.5 P0 + 0.5 P2

P’(1) = -0.5 P1 + 0.5 P3

B-Spline:

P(0) = ( P0 + 4 P1 + P2 ) / 6

P(1) = ( P1 + 4P2 + P3 ) / 6

P’(0) = -0.5 P0 + 0.5 P2

P’(1) = -0.5 P1 + 0.5 P3

Parts of the video:

22' - 23', improvement, to obtain the spline described in the previous pages.

38' Geometric continuity

43 Hermit Bezier

46' - 47' Cardinal spline

48' Catmull-Rom spline

54' Matrix of B-spline

57' Summary of the 4 splines above.

More precise definition of a B-spline and links with Math-splines

After having more precisely defined the B-spline curve obtained by control points, we will see how to pass from the control points of a B-spline to the control points of a Math ‑spline and vice versa.

Both curves are the same, just the way to control them using various points differ.

The advantage of a B-spline is that moving a control point only changes the 4 adjacent segments. Its disadvantage is that it does not pass through checkpoints.

The advantage of a Math-spline is to pass through all the control points, except the first and the last which we will define later. Its disadvantage is that it theoretically modifies all the segments of the curve, although practically it only modifies the 8 adjacent segments.

To avoid confusion with the letters used previously, here are notation conventions:

The control points of the B-spline will be denoted: *U i* , for *i* = 0 .. *n* +1.

The Math-spline control points will be denoted: *V i* , for *i* = 0 .. *n* +1.

The *U i* and the *V i* are points or vectors of dimension 2 for a plane and of higher dimension for a larger space. (3 for our space and more for a mathematician.)

The spline curve will be denoted *s* ( *t* ), for *t* = *t* 1 to *t* n , satisfying: , for *i* = 1 .. *n* .

It is defined by

to

is piecewise cubic.

Simplification for a first approach.

and so for all *i* .

Let us look for a matrix *A* making the spline curve twice continuously differentiable, to find the matrix linked to a B-spline.

For any *U i* , the conditions to be met are:

You can skip the calculations on the next page, to see only the result.

imposes the following condition, which breaks down into 5 conditions:

+

+

+

=

Since the equality must be true for all *U i* , it gives 5 equations.

imposes the following condition, which breaks down into 5 conditions:

+

+

+

=

Since the equality must be true for all *U i* , it gives 5 equations.

imposes the following condition, which breaks down into 5 conditions:

+

+

+

=

Since the equality must be true for all *U i* , it gives 5 equations.

We obtain 15 equations quite easy to solve, with a free unknown.

Concerning the *U i* +3 we obtain: ; ;

Concerning the *U i* +2 we obtain:

, So

, So

, So

Concerning the *U i* +1 we obtain:

, So

, So

, So

Concerning the *U i* we obtain:

, So

, So

, So

Concerning the *U i* -1 we obtain:

, So

, So

, So

By solving 3 equations, we get: and and

Which constraint must still be satisfied to fix ?

We obtain :

,

We find the matrix of the B-spline curve, with a free parameter, which corresponds to subjecting the whole curve to a homothety.

For that corresponds to a weighted average of the *U i* , we choose the value of . Another choice would make the curve dependent on the choice of plane origin. It is necessary that if we translate all the *U i* in the same way, that we calculate the B-spline curve, then that we translate it in the other direction, we obtain the same curve independently of the translation.

Thus we obtain exactly the characteristic matrix of a B-spline.

In appendix II, I recheck that the curve thus obtained is indeed twice continuously differentiable.

A first relation between the control points of the B-spline and those of the Math-spline:

, for *i* = *1..n* .

The two control points *V* 0 and *V* n+1 remain .

The Math-spline has two degrees of freedom corresponding to the slope of the curve at the start and the slope of the curve at the finish.

+

+

+

Derived from s(t).

+

+

Let's calculate the second derivative of s(t).

+

Determination of the two control points *V* 0 and *V* n+1 of the Math-spline to make the two curves coincide.

Slope at start = .

Slope on arrival = .

For the Math-spline, these two slopes are free to choose.

It remains to define a link between these two slopes and the two values *V* 0 and *V* n+1 .

We want the following conditions to be fulfilled:

1)

2) , with free

3)

4) , with free

We therefore want to express *V* 0 as a function of *U* 0 , *U* 1 and *U* 2 , which makes it possible to determine *p* 1 which satisfies condition 1), which describes the derivative at the start of the curve.

From these 4 equalities, let *p* 1 , *V* 0 and *V* 1 disappear .

1) = 4) and substitute 2) and 3), to obtain:

By rearranging the terms and highlighting the *U i* :

We want the equality to be true regardless of the values of *U i* , so we must:

and

and

A natural choice is: . With this choice, we get: and

; ; .

So :

*Parenthesis, not accepted.*

A pleasant choice to obtain a tri-diagonal matrix for passing from *U* i to *V* i would be to have , therefore and

and , , . For *p* 1 this is not natural. FRO.

Let's do similar calculations to determine *V* n+1 .

We want the following conditions to be fulfilled:

1)

2) , with free

3)

4) , with free

We therefore want to express *V n* +1 as a function of *U* n-1 , *U* n and *U* n+1 , which makes it possible to determine *p* n which satisfies condition 1), which describes the derivative at the end of the curve.

From these 4 equalities, let *p* n , *V* n and *V* n+1 disappear .

1) = 4) and substitute 2) and 3), to obtain:

By rearranging the terms and highlighting the *U i* :

We want the equality to be true regardless of the values of *U i* , so we must:

and

and

A natural choice is: . With this choice, we get: and

; ; .

So :

To calculate the curve of the B-spline passing through the control points:

+

+

+

Writing in matrix form of the transition from *U i* to *V i* .

=

Here, *n* = 4. There are 4 waypoints and 6 control points.

So the transition from the B-spline control points to the Math-spline control points is:

;

, for *i* = 1.. *n*

Passing from Math-spline control points to B-spline control points.

Going from Math-spline control points to B-spline control points requires solving the system of equations. Since the matrix is tri-diagonal with dominant diagonal, the calculation is quite fast. The *V i* are given, we seek the *U i* .

To get a tri-diagonal matrix, let's combine the first two rows and the last two.

=

Resolution :

; ;

, for *i* = 1.. *n*

, , , for *i* = 1.. *n*

and

for i=2 to *n* do and

and

for *i* = *n* downto 1 do There are *n* + 2 points; *n* = nb\_points - 2.

Out of curiosity, let's look at what has become of the system after triangulation by the Gaussian method described in the resolution above.

=   
6 \*

We notice that very quickly the diagonal of the matrix converges towards .

We also note that the *V i* quickly no longer influence the remote *U i .*

Closed B-splines.

In the case of the drawing, *n* = 7

*U*n+1

*U*3

*U*4

*s*1

*U*2

*s*3

*U*n

*U*0

*U*n-2

*s*2

*s*n-2

*s*n

*s*n-1

*U*n-1

*U*1

*s*4

*U*n+2

The goal is to add a segment *s n* , so that the curve is closed.

To do this, we will add a point *U n* +2 and the segment *s n* associated with the points *U n* +2 , *U n* +1 , *U n* and *U n* -1 , and allow the position of the points *U n* +1 to be modified , and *U* 0 .

We want this segment to end on *U* 0 , ie C 1 and C 2 at *U* 0 .

The conditions of continuity, of continuity of the derivative and of the second derivative are satisfied between *s n* ‑1 and *s n* if we have as usual:

+

+

+

On the other hand, three new conditions must be satisfied to have the continuity of the curve, the derivative and the second derivative.

It is therefore necessary to satisfy:

*s n* ( *t* n+1 ) = *s* 1 ( *t* 1 ), ( *t* n+1 = *t* n + 1). So and

*s' n* ( *t* n+1 ) = *s'* 1 ( *t* 1 ). So

*s'' n* ( *t* n+1 ) = *s''* 1 ( *t* 1 ). So

We check that they are satisfied if and only if:

and and

In conclusion, to close a B-spline, it is necessary to add a point *U n* +2 , place it on point *U* 2 and place point *U* 0 on *U* n and place point *U* n+1 on *U* 1 .

The closed Math-spline linked to the points *V* 0 to *V* n+1 corresponding to the points *U* 0 to *U* n+1 , will give the same curve as the closed B-spline described above.

Note that the closed B-spline has one more point than the closed Math-spline!

The modification of the position of the points *U* 0 and *U* n+1 will modify the position of the points *V* 0 , *V* 1 , *V* n and *V* n+1 , therefore the Math-spline curve. But it will not change the position of other points.

For a closed Math-spline, points *V* 0 and *V* n+1 are ignored.

These modifications of the points *U* 0 and *U* n+1 make it possible to close the curve while keeping it smooth, without disturbing the segments *s* 2 to *s n* -2 .

If a curve of a Math-spline is closed, then the corresponding B-spline will automatically have and . Closing the B-spline will give the same curve as the Math-spline.

Study of the general case, where the transit times *t i* are no longer regular.

Let us summarize the characteristics of B-splines of degree 3.

A B-spline (of degree 3) is characterized by the data of:

° control points of the B-spline will be denoted: *U i* , for *i* = 0 .. *n* +1;

° of passage time *t i* for *i* = 1 .. *n* and

° of *n* -1 4x4 characteristic matrices which will be denoted *A i*

such that if we define:

The curve (B-spline) defined by for ,   
is twice continuously differentiable.

The *U i* are points or vectors of dimension 2 for a plane and of higher dimension for a larger space. (3 for our space and more for a mathematician.)

To make the connection with the above: , for *i* = 1 .. *n* .

is therefore piecewise cubic.

*The question is how to determine the coefficients a i, j; k above so that the curve s(t) defined above is always twice continuously differentiable, whatever the points U i .*

This time, the coefficients a i, j; k will depend on *i* .

In order not to weigh down the writing, the *i -dependency* will not always be explicit.

*In the next few pages, there are a lot of calculations! They can be skipped, to see the result in the summary that follows these calculations.*

For this, let's start from the definition of a B-spline from the Wikipedia page:

[https://en.wikipedia.org/wiki/B-spline](https://fr.wikipedia.org/wiki/B-spline)

Let us limit the case of B-spline of degree 3, let us transform the notation to adapt it to that used previously.

Given *m* +1 nodes with   
a B-spline curve of degree 3 is a parametric curve *s* ( *t* ) defined as follows:

,   
where the *P i* form a polygon called the control polygon.

The number of points composing this polygon is equal to *m* - 3.

The *m* -3 B-spline functions of degree *k* are defined by induction on the lower degree:

, with

is zero if

Changed notations to match the one used previously.

;

*m* - 3 = *n* + 2 = number of control points

*n* = *m* -5; *m* = *n* + 5

So

,

, it remains to define the values of *t* -2 , *t* -1 and *t* 0 !?!

, with , so

is zero if

is zero if also: is zero if .

,

Thus we can clearly see that *s i* ( *t* ) only depends on the points *U* i-1 to *U* i+2 .

Let's explain them .

+

Let us evaluate in to already have coefficients.

Let's remember that :

and

So there is a direct link between the coefficients of the matrix and the

They are explained below.

Recall that is zero if .

only if *i* = *j* . In the case where , we have

+

First coefficient:

=

+

== \_

=

So

Verification in the equidistant case where , :

, that's what we've been waiting for!

Second coefficient:

Recall that =

+

=

+

== \_

=

=

So

Verification in the equidistant case where , :

, that's what we've been waiting for!

Third coefficient:

=

+

== \_

+ 0 =

So

Verification in the equidistant case where , :

, that's what we've been waiting for!

Fourth coefficient:

=

0 +

= 0

So that's what we expected

A more important check is:

If this were not the case, a translation of the control points would not result in a translation of the curve. It would therefore depend on the choice of origin.

Useful rule for simplifications:

It's magic ! That works !

Let's find the coefficients of the first column,

These are the coefficients of the polynomial of , written as a polynomial in .

Let's remember that :

+

and

So

is the factor of

is the factor of

is the factor of

is the factor of

+

, so only if *j* = *i* .

This simplifies:

= 0 +

=

- + - +

= - + - +

The 4 factors give the 4 coefficients: ; ; and

We find the value of

So we have :

= in the equidistant case where ,

= -3 in the equidistant case

= in the equidistant case

= in the equidistant case

Let's find the coefficients of the second column,

These are the coefficients of the polynomial of , written as a polynomial in .

Let's remember that :

+

and

So

is the factor of

is the factor of

is the factor of

is the factor of

+

, so only if *j* = *i* .

This simplifies:

+

+

+

+

+

The 4 factors give the 4 coefficients: ; ; and

We find the value of

So we have :

= in the equidistant case where ,

= 0 if equidistant

= in the equidistant case

= in the equidistant case

Let's find the coefficients of the third column,

These are the coefficients of the polynomial of , written as a polynomial in .

Let's remember that :

+

and

So

is the factor of

is the factor of

is the factor of

is the factor of

+

, so only if *j* = *i* .

This simplifies:

+

-

-

-

+

+

+

The 4 factors give the 4 coefficients: ; ; and

We find the value of

So we have :

= in the equidistant case where ,

= if equidistant

= in the equidistant case

= in the equidistant case

Let's find the coefficients of the fourth column,

These are the coefficients of the polynomial of , written as a polynomial in .

Let's remember that :

+

and

, so only if *j* = *i* .

This simplifies:

is the factor of of , so

= in the equidistant case where ,

; ;

**Summary - result - conclusion of previous calculations.**

with

The curve (B-spline) defined by for ,   
, so

*A i* =

If we want to have

We have the relation: *i* = *1..n* ; *j* = 1..4; *k* = 1 .. 4

For *i* = 1 .. *n* , we have:

The relation between the *V i* and the *U i* can be written as a matrix, which makes it possible to pass control points from the B-spline to the Math-spline and vice versa.

How to determine the two control points *V* 0 and *V* n+1 of the Math-spline to make the two curves coincide?

Slope at start = .

Slope on arrival = .

is only defined at *t* = *t n* and is easier to compute than .

For the Math-spline, these two slopes are free to choose.

It remains to define a link between these two slopes and the two values *V* 0 and *V* n+1 .

We want the following conditions to be fulfilled:

1)

2) , with free

3)

4) , with free

We therefore want to express *V* 0 as a function of *U* 0 , *U* 1 and *U* 2 , which makes it possible to determine *p* 1 which satisfies condition 1), which describes the derivative at the start of the curve.

From these 4 equalities, let *p* 1 , *V* 0 and *V* 1 disappear .

1) = 4) and substitute 2) and 3), to obtain:

By rearranging the terms and highlighting the *U i* :

We want the equality to be true regardless of the values of *U i* , so we must:

and

and

A natural choice is: . With this choice, we get: and

; ; .

So :

Let's do similar calculations to determine *V* n+1 .

We want the following conditions to be fulfilled:

1)

2) , with free

3)

4) , with free

We therefore want to express *V n* +1 as a function of *U* n-1 , *U* n and *U* n+1 , which makes it possible to determine *p* n which satisfies condition 1), which describes the derivative at the end of the curve.

From these 4 equalities, let *p* n , *V* n and *V* n+1 disappear .

1) = 4) and substitute 2) and 3), to obtain:

By rearranging the terms and highlighting the *U i* :

We want the equality to be true regardless of the values of *U i* , so we must:

and

and

A natural choice is: . With this choice, we get: and

; ; .

So :

Writing in matrix form of the transition from *U i* to *V i* .

=

Here, *n* = 5. There are 5 waypoints and 7 checkpoints.

; ;

; ;

So the transition from the B-spline control points to the Math-spline control points is:

;

, for *i* = 1.. *n*

Passing from Math-spline control points to B-spline control points.

Going from Math-spline control points to B-spline control points requires solving the system of equations. Since the matrix is tri-diagonal with dominant diagonal, the calculation is quite fast. The *V i* are given, we seek the *U i* .

To get a tri-diagonal matrix, let's combine the first two rows and the last two.

Substitutions:

; ; ;

; ; ;

Resolution :

; ; ;

; ; ; (substituted values)

; ; ; , for *i* = 1.. *n*

for i=1 to *n* +1 do and

for *i* = *n* downto 0 do There are *n* + 2 points; *n* = nb\_points - 2.

Verifications!

It is unlikely to do all the calculations on the previous pages without making mistakes. It is therefore important to make several checks, which will allow errors to be detected.

Similarly, during the computer implementation, the following checks will make it possible to detect coding errors.

We want a translation of the control points to give the same translated curve.

Otherwise, the curve would be dependent on the choice of origin.

This implies the following 4 equalities:

1) , verification that has already been done.

2)

3)

4)

Useful rule for simplifications:

;

2) gives

+

=

+

=

+

=

+

=

+

=

+

=

+

= +

=

=

= 0. It's long, but it's magic.

There is little chance of having this result with an error in the coefficients of the matrix!

3)

“This verification is left as an exercise. We say that when we don't want to do it ourselves.

4) , this verification is easy.

+ = 0

Another approach is attempted in the following pages. It does not lead to a final result, but gives several possible verifications.

Let us look for a matrix *A* making the spline curve twice continuously differentiable, to find the matrix linked to a B-spline.

U

The *a* j; k will depend on the *h i* .

The *bj* ; k must depend on the *h i* in the same way as the *a* j; k with the value of *i* shifted by +1.

For any *U i* , the conditions to be met are:

The equality must be true all *U i* , it imposes conditions, which breaks down into 5 equations:

, so also : which has been seen previously.

The first condition is easily verified:

== 0

The other conditions are more complicated to verify.

The equality must be true all *U i* , it imposes conditions, which breaks down into 5 equations:

, so also : which has been seen previously.

The first condition is easily verified:

== 0

The other conditions are more complicated to verify.

The equality must be true all *U i* , it imposes conditions, which breaks down into 5 equations:

, so also : which has been seen previously.

The first condition is easily verified:

== 0

The other conditions are more complicated to verify.

They can be verified numerically on examples, this can be useful for verifying the program.

Closed B-splines in the case of non-regular times *t i .*

In the case of the drawing, *n* = 7

*U*n+1 =

*U*3

*U*4

*s*1

*U*2

*s*3

*U*n

*U*0 =

*U*n-2

*s*2

*s*n-2

*s*n

*s*n-1

*U*n-1

*U*1

*s*4

*U*n+2 =

The goal is to add a segment *s n* , so that the curve is closed.

To do this, we will add a point *U n* +2 and the segment *s n* associated with the points *U n* +2 , *U n* +1 , *U n* and *U n* -1 , and allow the position of the points *U n* +1 to be modified , and *U* 0 .

We want this segment to end on *U* 0 , ie C 1 and C 2 at *U* 0 .

The conditions of continuity, of continuity of the derivative and of the second derivative are satisfied between *s n* ‑1 and *s n* if we have as usual:

On the other hand, three new conditions must be satisfied to have the continuity of the curve, the derivative and the second derivative.

It is therefore necessary to satisfy:

*s n* ( *t* n+1 ) = *s* 1 ( *t* 1 ), ( *t* n+1 = *t* n + 1). So

and

*s' n* ( *t* n+1 ) = *s'* 1 ( *t* 1 ). So

*s'' n* ( *t* n+1 ) = *s''* 1 ( *t* 1 ). So

In the case where one has chosen: , with for example = distance between *U* 1 and *U n* .

We check that they are satisfied if and only if:

and and

In conclusion, to close a B-spline, it is necessary to add a virtual point *U n* +2 , place it on point *U* 2 and place point *U* 0 on *U* n and place point *U* n+1 on *U* 1 .

The closed Math-spline linked to the points *V* 0 to *V* n+1 corresponding to the points *U* 0 to *U* n+1 , will give the same curve as the closed B-spline described above.

Note that the closed B-spline has one more point than the closed Math-spline!

The modification of the position of the points *U* 0 and *U* n+1 will modify the position of the points *V* 0 , *V* 1 , *V* n and *V* n+1 , therefore the Math-spline curve. But it will not change the position of other points.

For a closed Math-spline, points *V* 0 and *V* n+1 are ignored.

These modifications of the points *U* 0 and *U* n+1 make it possible to close the curve while keeping it smooth, without disturbing the segments *s* 2 to *s n* -2 .

If a curve of a Math-spline is closed, then the corresponding B-spline will automatically have and . Closing the B-spline will give the same curve as the Math-spline.

Table of different cases of comparisons of B-splines and Math-splines

Let's compare various situations of links between B-splines and Math-splines

**dT = 1**

B-spline Ouvert Fermé

Math-spline

B => M OK B => M OK

Ouvert M => B OK M => B X

B => M X B => M OK

Fermé M => B OK M => B OK

**dT = distance entre points**

B-spline Ouvert Fermé

Math-spline

B => M OK B => M OK

Ouvert M => B OK M => B X

B => M X B => M OK

Fermé M => B OK M => B OK

"B => M" means: "move a point of the B-spline and update the points of the Math-spline"

"M => B" means: "move a point of the Math-spline and update the points of the B-spline"

OK means curves match after point move.

X means that the curves no longer necessarily match after the point move.

Case of open B-spline and closed Math-spline:

° it is normal that moving a point of the B-spline does not make it possible to have correspondence of the curves, because for a closed curve, points of the B- ‑spline must overlap, which is not imposed when the B-spline is open.

° moving a point of the Math-spline will force the overlapping points of the B-spline to give a curve superimposed on that of the Math-spline.

Case of closed B-spline and open Math-spline:

° it is normal that moving a point of the Math-spline does not make it possible to have correspondence of the curves, because for a closed curve, the first and the last point of the Math- ‑spline are imposed, which is not the case when the Math-spline is open.

° moving a point of the B-spline will force the positioning of the first and last point of the Math ‑spline so that it gives a curve superimposed on that of the B-spline.

In fact, it will also force a positioning of the second and penultimate point.

In the case where both are open, there are no opposites which limit the adaptation of one curve to the other.

In the case where both are closed, both undergo constraints which allow the superposition of one curve on the other.

Appendix I, approximation of the second derivative of a function from 3 points.

To calculate a periodic spline curve, we approximated the second derivative at the start and end of the curve. The following justifies these approximations.

Let *f be* a C 2 function , hence twice continuously differentiable on the interval [ *t m* ; *p* ] *\_*

*t m* < *t* 0 < *t p* given.

With: and , both positive.

We have :

where

where

Let us show that the following expression is an approximation of the second derivative at *t* 0 .

which is a weighted average of the second derivative before *t* 0 and after *t* 0 .

This fraction is greater than and   
less than

Since the second derivative is continuous, the fraction is therefore equal to   
for one .

It is therefore an approximation of the second derivative at *t* 0 .

Annex II.

Checks that the curve obtained by B-spline is twice continuously differentiable.

Let's remember that :

So

+

+

+

Let's calculate the derivative of s(t).

+

+

Let's calculate the second derivative of s(t).

+

Verification of continuity in *t i* .

okay

Verification of the continuity of the derivative in *t i* .

okay

Checking the continuity of the second derivative at *t i* .

okay