WIKIPEDIA

Cubic Hermite spline

In numerical analysis, a **cubic Hermite spline** or **cubic Hermite interpolator** is a spline where each piece is a third-degree polynomial specified in Hermite form:[1] i.e., by its values and first derivatives at the end points of the corresponding domain interval.

Cubic Hermite splines are typically used for interpolation of numeric data specified at given argument values $x_1, x_2, \ldots, x_n$, to obtain a smooth continuous function. The data should consist of the desired function value and derivative at each x_k. (If only the values are provided, the derivatives must be estimated from them.) The Hermite formula is applied to each interval (x_k, x_{k+1}) separately. The resulting spline will be continuous and will have continuous first derivative.

Cubic polynomial splines can be specified in other ways, the Bézier form being the most common. However, these two methods provide the same set of splines, and data can be easily converted between the Bézier and Hermite forms; so the names are often used as if they were synonymous.

Cubic polynomial splines are extensively used in computer graphics and geometric modeling to obtain curves or motion trajectories that pass through specified points of the plane or three-dimensional space. In these applications, each coordinate of the plane or space is separately interpolated by a cubic spline function of a separate parameter *t*. Cubic polynomial splines are also used extensively in structural analysis applications, such as Euler–Bernoulli beam theory.

Cubic splines can be extended to functions of two or more parameters, in several ways. Bicubic splines (Bicubic interpolation) are often used to interpolate data on a regular rectangular grid, such as pixel values in a digital image or altitude data on a terrain. Bicubic surface patches, defined by three bicubic splines, are an essential tool in computer graphics.

Cubic splines are often called **csplines**, especially in computer graphics. Hermite splines are named after Charles Hermite.

Contents

Interpolation on a single interval

Unit interval (0, 1)

On the unit interval $(0,1)$, given a starting point $\boldsymbol{p}_0$ at $t=0$ and an ending point $\boldsymbol{p}_1$ at $t=1$ with starting tangent $\boldsymbol{m}_0$ at $t=0$ and ending tangent $\boldsymbol{m}_1$ at $t=1$, the polynomial can be defined by

$$\boldsymbol{p}(t) = (2t^3 - 3t^2 + 1)\boldsymbol{p}_0 + (t^3 - 2t^2 + t)\boldsymbol{m}_0 + (-2t^3 + 3t^2)\boldsymbol{p}_1 + (t^3 - t^2)\boldsymbol{m}_1$$

where $t \in [0, 1]$.

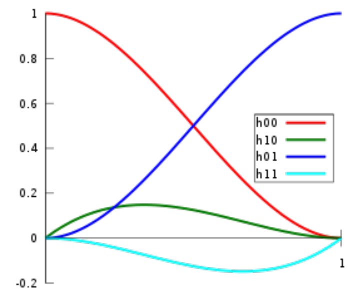
Interpolation on an arbitrary interval

Interpolating x in an arbitrary interval (x_k, x_{k+1}) is done by mapping the latter to $[0,1]$ through an affine (degree 1) change of variable. The formula is

$$\boldsymbol{p}(x) = h_{00}(t)\boldsymbol{p}_k + h_{10}(t)(x_{k+1} - x_k)\boldsymbol{m}_k + h_{01}(t)\boldsymbol{p}_{k+1} + h_{11}(t)(x_{k+1} - x_k)\boldsymbol{m}_{k+1}.$$

with $t = (x - x_k)/(x_{k+1} - x_k)$ and h refers to the basis functions, defined below. Note that the tangent values have been scaled by $x_{k+1} - x_k$ compared to the equation on the unit interval.

Uniqueness



The four Hermite basis functions. The interpolant in each subinterval is a linear combination of these four functions.

The formulae specified above provide the unique third-degree polynomial path between the two points with the given tangents.

Proof. Let P, Q be two third degree polynomials satisfying the given boundary conditions. Define $R = Q - P$, then:

$$R(0) = Q(0) - P(0) = 0,$$
$$R(1) = Q(1) - P(1) = 0.$$

Since both Q and P are third degree polynomials, R is at most a third degree polynomial. So R must be of the form:

$$R(x) = ax(x-1)(x-r),$$

calculating the derivative gives:

$$R'(x) = ax(x-1) + ax(x-r) + a(x-1)(x-r).$$

We know furthermore that:

$$R'(0) = Q'(0) - P'(0) = 0$$

$$R'(0) = 0 = ar \qquad (1)$$

$$R'(1) = Q'(1) - P'(1) = 0$$

$$R'(1) = 0 = a(1-r) \qquad (2)$$

Putting (**1**) and (**2**) together, we deduce that $a = 0$ and therefore $R = 0$, thus $P = Q$.

Representations

We can write the interpolation polynomial as

$$\boldsymbol{p}(t) = h_{00}(t)\boldsymbol{p}_0 + h_{10}(t)\boldsymbol{m}_0 + h_{01}(t)\boldsymbol{p}_1 + h_{11}(t)\boldsymbol{m}_1$$

where h_{00}, h_{10}, h_{01}, h_{11} are Hermite basis functions. These can be written in different ways, each way revealing different properties.

	expanded	factorized	Bernstein
$h_{00}(t)$	$2t^3 - 3t^2 + 1$	$(1+2t)(1-t)^2$	$B_0(t) + B_1(t)$
$h_{10}(t)$	$t^3 - 2t^2 + t$	$t(1-t)^2$	$\frac{1}{3} \cdot B_1(t)$
$h_{01}(t)$	$-2t^3 + 3t^2$	$t^2(3-2t)$	$B_3(t) + B_2(t)$
$h_{11}(t)$	$t^3 - t^2$	$t^2(t-1)$	$-\frac{1}{3} \cdot B_2(t)$

The "expanded" column shows the representation used in the definition above. The "factorized" column shows immediately, that h_{10} and h_{11} are zero at the boundaries. You can further conclude that h_{01} and h_{11} have a zero of multiplicity 2 at 0 and h_{00} and h_{10} have such a zero at 1, thus they have slope 0 at those boundaries. The "Bernstein" column shows the decomposition of the Hermite basis functions into Bernstein polynomials of order 3:

$$B_k(t) = \binom{3}{k} \cdot t^k \cdot (1-t)^{3-k}$$

Using this connection you can express cubic Hermite interpolation in terms of cubic Bézier curves with respect to the four values $\boldsymbol{p}_0, \boldsymbol{p}_0 + \frac{\boldsymbol{m}_0}{3}, \boldsymbol{p}_1 - \frac{\boldsymbol{m}_1}{3}, \boldsymbol{p}_1$ and do Hermite interpolation using the de Casteljau algorithm. It shows that in a cubic Bézier patch the two control points in the middle determine the tangents of the interpolation curve at the respective outer points.

Interpolating a data set

A data set, $(t_k, \boldsymbol{p}_k)$ for $k = 1, \ldots, n$, can be interpolated by applying the above procedure on each interval, where the tangents are chosen in a sensible manner, meaning that the tangents for intervals sharing endpoints are equal. The interpolated curve then consists of piecewise cubic Hermite splines, and is globally continuously differentiable in (t_1, t_n).

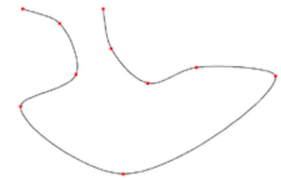
The choice of tangents is non-unique, and there are several options available.

Finite difference

The simplest choice is the three-point difference, not requiring constant interval lengths,

$$\boldsymbol{m}_k = \frac{1}{2}\left(\frac{\boldsymbol{p}_{k+1} - \boldsymbol{p}_k}{t_{k+1} - t_k} + \frac{\boldsymbol{p}_k - \boldsymbol{p}_{k-1}}{t_k - t_{k-1}}\right)$$

for internal points $k = 2, \dots, n-1$, and one-sided difference at the endpoints of the data set.



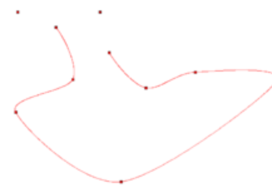
Example with finite difference tangents

Cardinal spline

A **cardinal spline**, sometimes called a **canonical spline**,[2] is obtained[3] if

$$\boldsymbol{m}_k = (1-c)\frac{\boldsymbol{p}_{k+1} - \boldsymbol{p}_{k-1}}{t_{k+1} - t_{k-1}}$$

is used to calculate the tangents. The parameter *c* is a *tension* parameter that must be in the interval [0,1]. In some sense, this can be interpreted as the "length" of the tangent. Choosing *c*=1 yields all zero tangents, and choosing *c*=0 yields a Catmull–Rom spline.



Cardinal spline example in 2D. The line represents the curve, and the squares represent the control points $\boldsymbol{p}_k$. Notice that the curve does not reach the first and last points; these points do however affect the shape of the curve. The tension parameter used is 0.1

Catmull–Rom spline

For tangents chosen to be

$$\boldsymbol{m}_k = \frac{\boldsymbol{p}_{k+1} - \boldsymbol{p}_{k-1}}{t_{k+1} - t_{k-1}}$$

a **Catmull–Rom spline** is obtained, being a special case of a cardinal spline. This assumes uniform parameter spacing.

The curve is named after Edwin Catmull and Raphael Rom. The principal advantage of this technique is that the points along the original set of points also make up the control points for the spline curve.[4] Two additional points are required on either end of the curve. The default implementation of the Catmull–Rom algorithm can produce loops and self intersections. The chordal and centripetal Catmull–Rom implementations [5] solve this problem, but use a slightly different calculation.[6] In computer graphics, Catmull–Rom splines are frequently used to get smooth interpolated motion between key frames. For example, most camera path animations generated from discrete key-frames are handled using Catmull–Rom splines. They are popular mainly for being relatively easy to compute, guaranteeing that each key frame position will be hit exactly, and also guaranteeing that the tangents of the generated curve are continuous over multiple segments.

Kochanek–Bartels spline

A Kochanek–Bartels spline is a further generalization on how to choose the tangents given the data points $\boldsymbol{p}_{k-1}$, $\boldsymbol{p}_k$ and $\boldsymbol{p}_{k+1}$, with three parameters possible, tension, bias and a continuity parameter.

Monotone cubic interpolation

If a cubic Hermite spline of any of the above listed types is used for interpolation of a monotonic data set, the interpolated function will not necessarily be monotonic, but monotonicity can be preserved by adjusting the tangents.

Interpolation on the unit interval with matched derivatives at endpoints

Considering a single coordinate of the points $\boldsymbol{p}_{n-1}, \boldsymbol{p}_n, \boldsymbol{p}_{n+1}$ and $\boldsymbol{p}_{n+2}$ as the values that a function, *f*(*x*), takes at integer ordinates *x*=*n*−1, *n*, *n*+1 and *n*+2,

$$p_n = f(n) \quad \forall n \in \mathbb{Z}$$

If, in addition, the tangents at the endpoints are defined as the centered differences of the adjacent points,

$$m_n = \frac{f(n+1) - f(n-1)}{2} = \frac{p_{n+1} - p_{n-1}}{2} \quad \forall n \in \mathbb{Z}$$

To evaluate the interpolated *f*(*x*) for a real *x*, first separate *x* into the integer portion, *n*, and fractional portion, *u*

$$\begin{aligned} &x = n + u \\ &n = \lfloor x \rfloor = \operatorname{floor}(x) \\ &u = x - n = x - \lfloor x \rfloor \\ &0 \le u < 1 \end{aligned}$$

Then the Catmull–Rom spline is [7]

$$
\begin{aligned}
f(x) = f(n+u) &= \mathrm{CINT}_u\left(p_{n-1}, p_n, p_{n+1}, p_{n+2}\right) \\
&= \begin{bmatrix} 1 & u & u^2 & u^3 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 1 & -\frac{5}{2} & 2 & -\frac{1}{2} \\ -\frac{1}{2} & \frac{3}{2} & -\frac{3}{2} & \frac{1}{2} \end{bmatrix} \cdot \begin{bmatrix} p_{n-1} \\ p_n \\ p_{n+1} \\ p_{n+2} \end{bmatrix} \\
&= \frac{1}{2} \begin{bmatrix} -u^3 + 2u^2 - u \\ 3u^3 - 5u^2 + 2 \\ -3u^3 + 4u^2 + u \\ u^3 - u^2 \end{bmatrix}^{\mathrm{T}} \cdot \begin{bmatrix} p_{n-1} \\ p_n \\ p_{n+1} \\ p_{n+2} \end{bmatrix} \\
&= \frac{1}{2} \begin{bmatrix} u((2-u)u-1) \\ u^2(3u-5)+2 \\ u((4-3u)u+1) \\ u^2(u-1) \end{bmatrix}^{\mathrm{T}} \cdot \begin{bmatrix} p_{n-1} \\ p_n \\ p_{n+1} \\ p_{n+2} \end{bmatrix} \\
&= \tfrac{1}{2}\Big(\big(u^2(2-u)-u\big)p_{n-1} \;+\; \big(u^2(3u-5)+2\big)p_n +\; \big(u^2(4-3u)+u\big)p_{n+1} \;+\; u^2(u-1)p_{n+2}\Big) \\
&= \tfrac{1}{2}\Big((-u^3+2u^2-u)p_{n-1} \;+\; (3u^3-5u^2+2)p_n +\; (-3u^3+4u^2+u)p_{n+1} \;+\; (u^3-u^2)p_{n+2}\Big) \\
&= \tfrac{1}{2}\Big((-p_{n-1}+3p_n-3p_{n+1}+p_{n+2})u^3 \;+\; (2p_{n-1}-5p_n+4p_{n+1}-p_{n+2})u^2 \;+\; (-p_{n-1}+p_{n+1})u \;+\; 2p_n\Big) \\
&= \tfrac{1}{2}\Big(\Big((-p_{n-1}+3p_n-3p_{n+1}+p_{n+2})u \;+\; (2p_{n-1}-5p_n+4p_{n+1}-p_{n+2})\Big)u \;+\; (-p_{n-1}+p_{n+1})\Big)u \;+\; p_n
\end{aligned}
$$

$\lfloor x \rfloor$ denotes the floor function which returns the largest integer no larger than x and $\mathbf{T}$ denotes the matrix transpose. The bottom equality is depicting the application of Horner's method.

This writing is relevant for tricubic interpolation, where one optimization requires you to compute CINT_u sixteen times with the same u and different p.

See also

- Bicubic interpolation, a generalization to two dimensions
- Tricubic interpolation, a generalization to three dimensions
- Hermite interpolation
- Multivariate interpolation
- Spline interpolation
- Discrete spline interpolation

References

1. Erwin Kreyszig (2005). *Advanced Engineering Mathematics* (9 ed.). Wiley. p. 816. ISBN 9780471488859.
2. Petzold, Charles (2009). "Canonical Splines in WPF and Silverlight" (http://www.charlespetzold.com/blog/2009/01/Canonical-Splines-in-WPF-and-Silverlight.html).
3. "Cardinal Splines" (http://msdn2.microsoft.com/en-us/library/ms536358.aspx). *Microsoft Developer Network*. Retrieved 2018-05-27.
4. Catmull, Edwin; Rom, Raphael (1974), "A class of local interpolating splines", in Barnhill, R. E.; Riesenfeld, R. F. (eds.), *Computer Aided Geometric Design*, New York: Academic Press, pp. 317–326
5. N. Dyn, M. S. Floater, and K. Hormann. Four-point curve subdivision based on iterated chordal and centripetal parameterizations. Computer Aided Geometric Design, 26(3):279{286, 2009
6. P. J. Barry and R. N. Goldman. A recursive evaluation algorithm for a class of Catmull-Rom splines. SIGGRAPH Computer Graphics, 22(4):199{204, 1988.
7. Two hierarchies of spline interpolations. Practical algorithms for multivariate higher order splines (https://arxiv.org/abs/0905.3564)

External links

- Spline Curves (http://www.cs.clemson.edu/~dhouse/courses/405/notes/splines.pdf), Prof. Donald H. House Clemson University
- Multi-dimensional Hermite Interpolation and Approximation (http://cvcweb.ices.utexas.edu/ccv/papers/1993/conference/multidim.pdf), Prof. Chandrajit Bajaj, Purdue University
- Introduction to Catmull–Rom Splines (http://www.mvps.org/directx/articles/catmull/), MVPs.org
- Interpolating Cardinal and Catmull–Rom splines (http://www.ibiblio.org/e-notes/Splines/Cardinal.htm)
- Interpolation methods: linear, cosine, cubic and hermite (with C sources) (http://paulbourke.net/miscellaneous/interpolation/)
- Common Spline Equations (http://www.blackpawn.com/texts/splines/)