SPLINE INTERPOLATION

Rodrigo Silveira

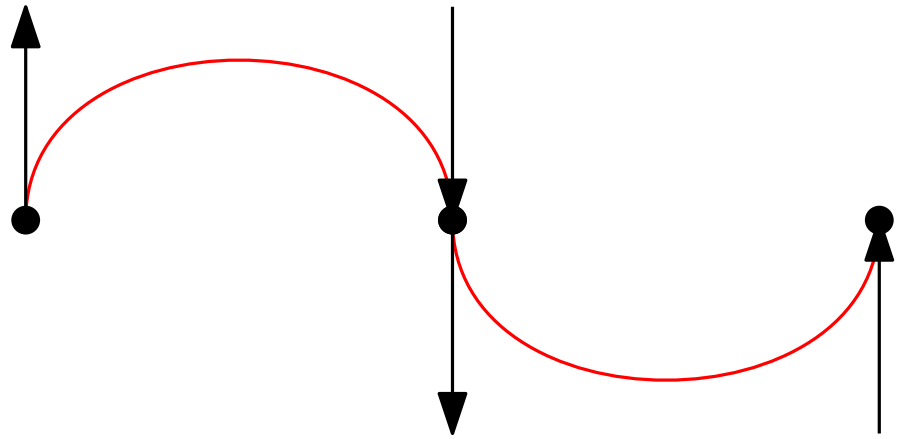
Curve and Surface Design
Facultat d'Informàtica de Barcelona
Universitat Politècnica de Catalunya

What can you do with Hermite curves? Combine them!

- A single Hermite curve is not so useful in practice, but it gets more interesting if you can put together several of them!

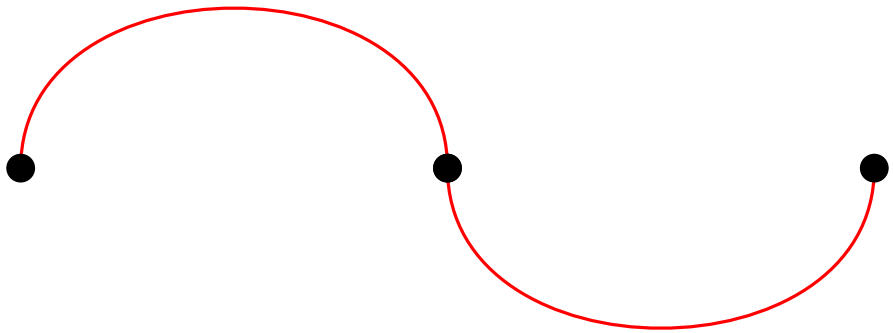
What can you do with Hermite curves? Combine them!

- A single Hermite curve is not so useful in practice, but it gets more interesting if you can put together several of them!



What can you do with Hermite curves? Combine them!

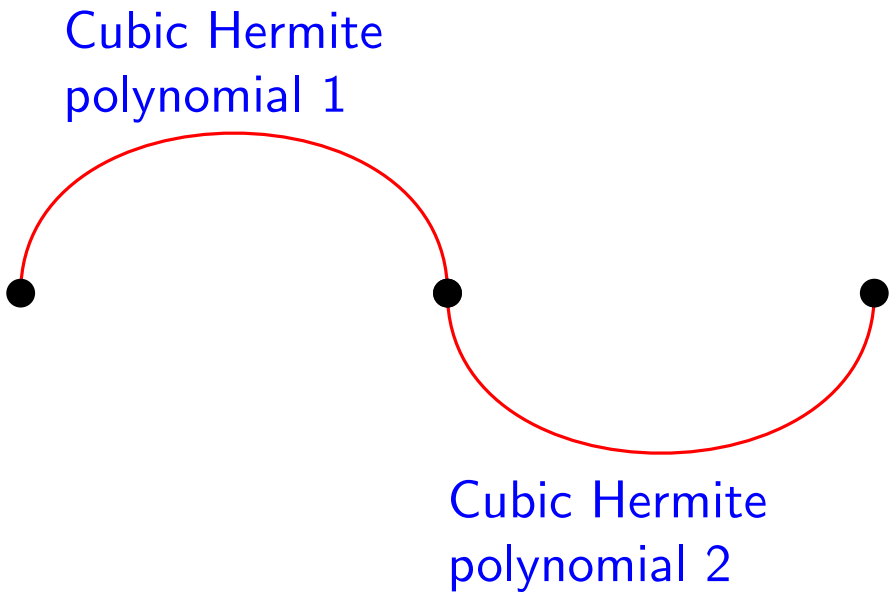
- A single Hermite curve is not so useful in practice, but it gets more interesting if you can put together several of them!



- **Spline**: set of polynomial curves of degree k that are smoothly connected at data points

What can you do with Hermite curves? Combine them!

- A single Hermite curve is not so useful in practice, but it gets more interesting if you can put together several of them!

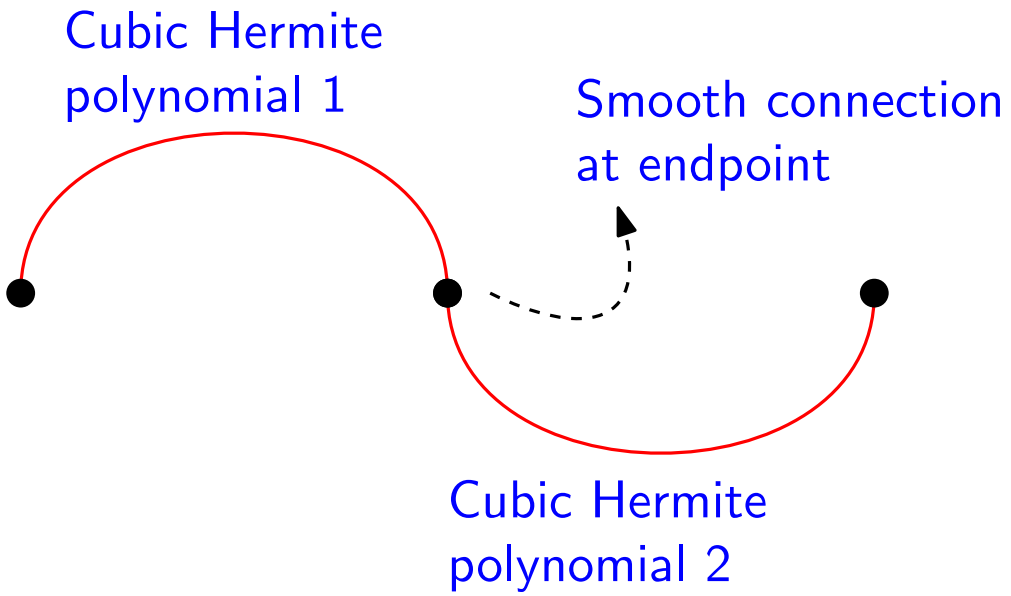


- **Spline**: set of polynomial curves of degree k that are smoothly connected at data points

SPLINE INTERPOLATION

What can you do with Hermite curves? Combine them!

- A single Hermite curve is not so useful in practice, but it gets more interesting if you can put together several of them!

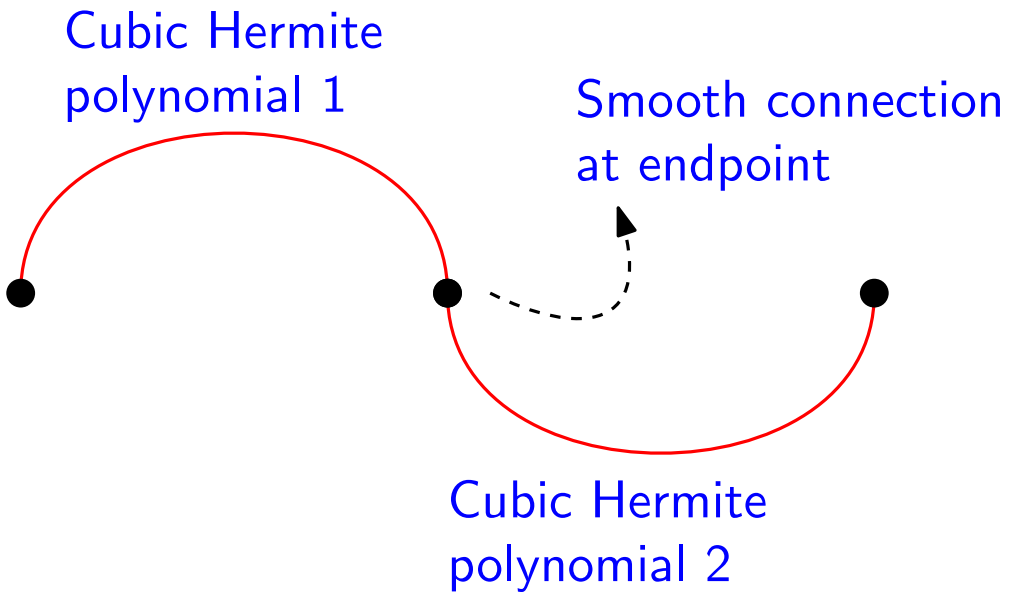


- **Spline**: set of polynomial curves of degree k that are smoothly connected at data points

SPLINE INTERPOLATION

What can you do with Hermite curves? Combine them!

- A single Hermite curve is not so useful in practice, but it gets more interesting if you can put together several of them!

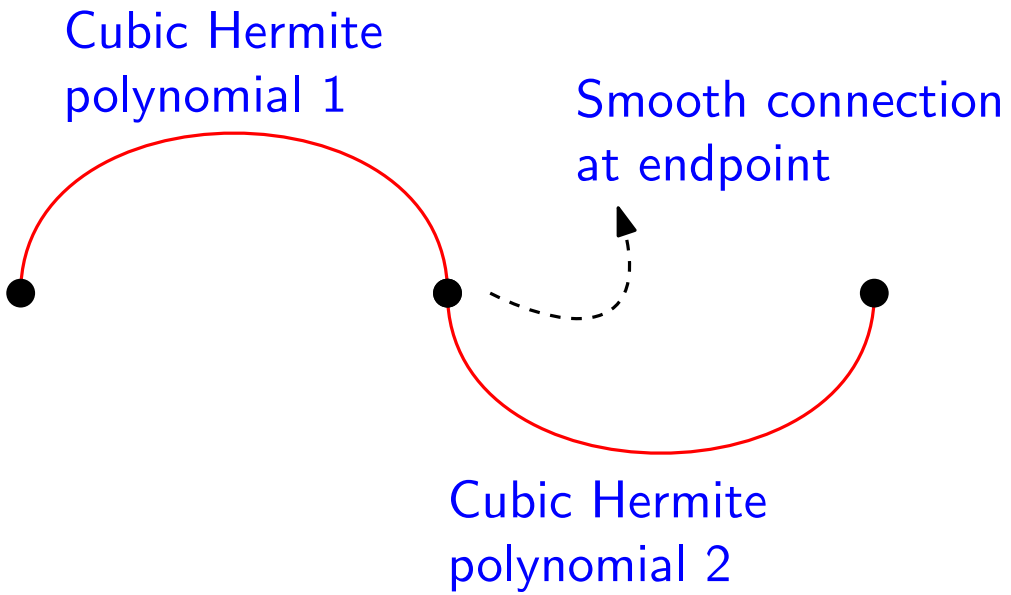


- **Spline**: set of polynomial curves of degree k that are smoothly connected at data points
- More precisely: at each data point (except for the first and last), two polynomials connect, and their derivatives up to the $(k-1)$th have the same values

SPLINE INTERPOLATION

What can you do with Hermite curves? Combine them!

- A single Hermite curve is not so useful in practice, but it gets more interesting if you can put together several of them!



- **Spline**: set of polynomial curves of degree k that are smoothly connected at data points
- More precisely: at each data point (except for the first and last), two polynomials connect, and their derivatives up to the $(k-1)$th have the same values
- Splines allow keeping the degree low, which in turn allows avoiding the wiggling of high-degree polynomials, as well as the complexity of their computation

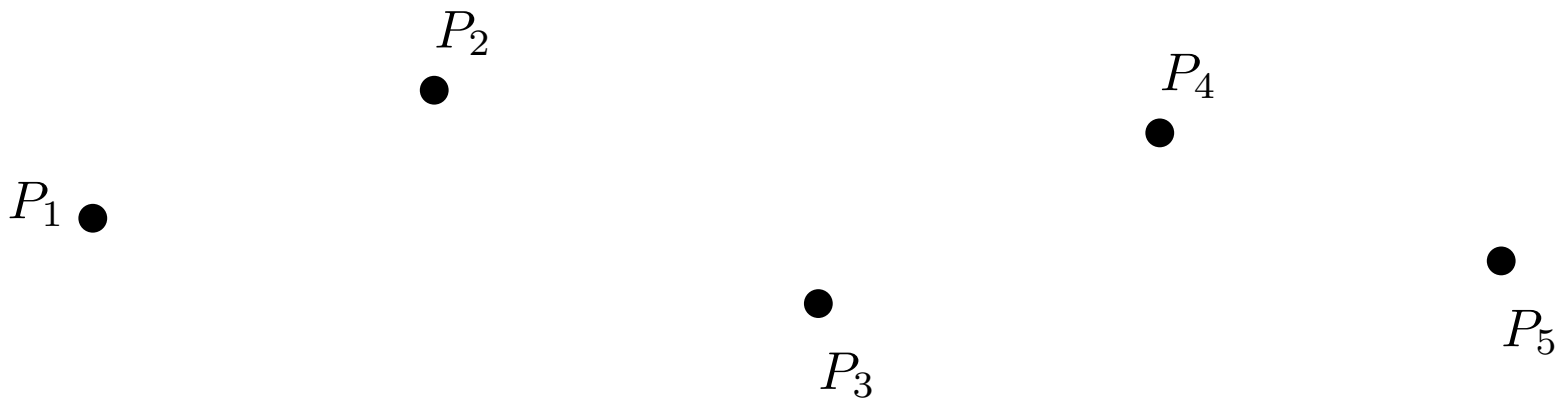
Cubic splines

- Input: n points $P_1, \ldots, P_n$ and 2 tangent vectors $\overrightarrow{v_1}$, $\overrightarrow{v_n}$ (other possibilities exist, see later)
- Output: A curve consisting of $n-1$ cubic Hermite curves that are smoothly connected (up to 2nd derivative) at the intermediate points $P_2, \ldots, P_{n-1}$

Cubic splines

- Input: n points $P_1, \ldots, P_n$ and 2 tangent vectors $\overrightarrow{v_1}$, $\overrightarrow{v_n}$ (other possibilities exist, see later)
- Output: A curve consisting of $n-1$ cubic Hermite curves that are smoothly connected (up to 2nd derivative) at the intermediate points $P_2, \ldots, P_{n-1}$

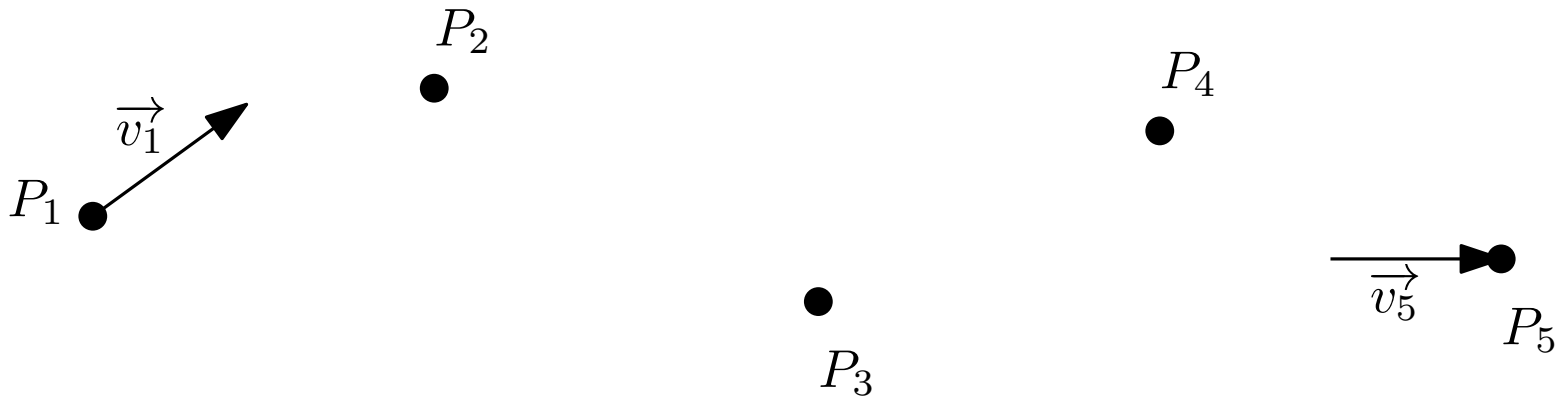
Example with $n = 5$



Cubic splines

- Input: n points $P_1, \ldots, P_n$ and 2 tangent vectors $\overrightarrow{v_1}$, $\overrightarrow{v_n}$ (other possibilities exist, see later)
- Output: A curve consisting of $n-1$ cubic Hermite curves that are smoothly connected (up to 2nd derivative) at the intermediate points $P_2, \ldots, P_{n-1}$

Example with $n = 5$

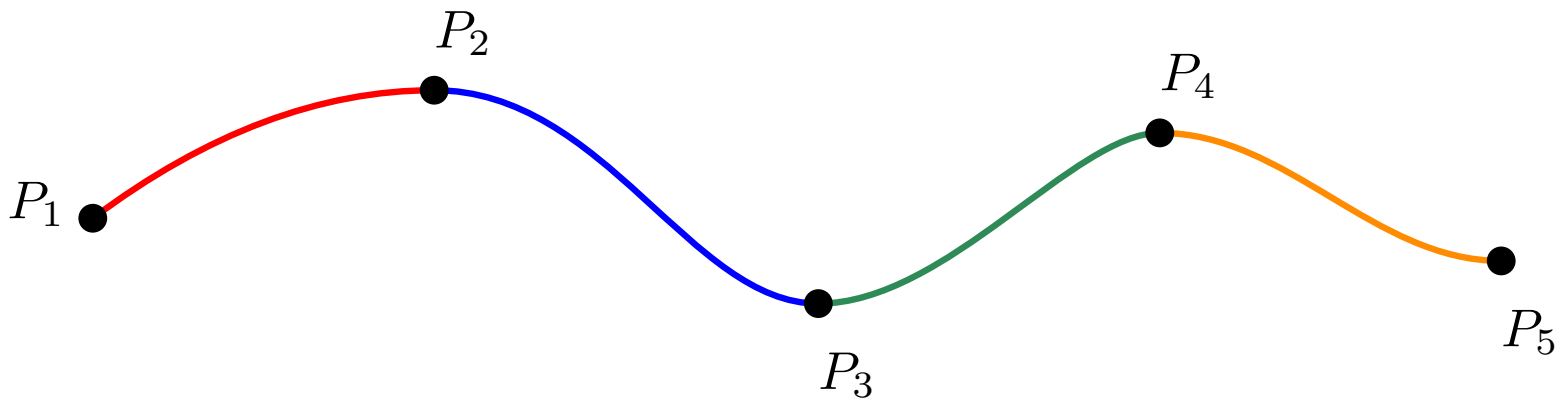


Input: $P_1, \ldots, P_5$ and $\overrightarrow{v_1}, \overrightarrow{v_5}$

Cubic splines

- Input: n points $P_1, \ldots, P_n$ and 2 tangent vectors $\overrightarrow{v_1}$, $\overrightarrow{v_n}$ (other possibilities exist, see later)
- Output: A curve consisting of $n-1$ cubic Hermite curves that are smoothly connected (up to 2nd derivative) at the intermediate points $P_2, \ldots, P_{n-1}$

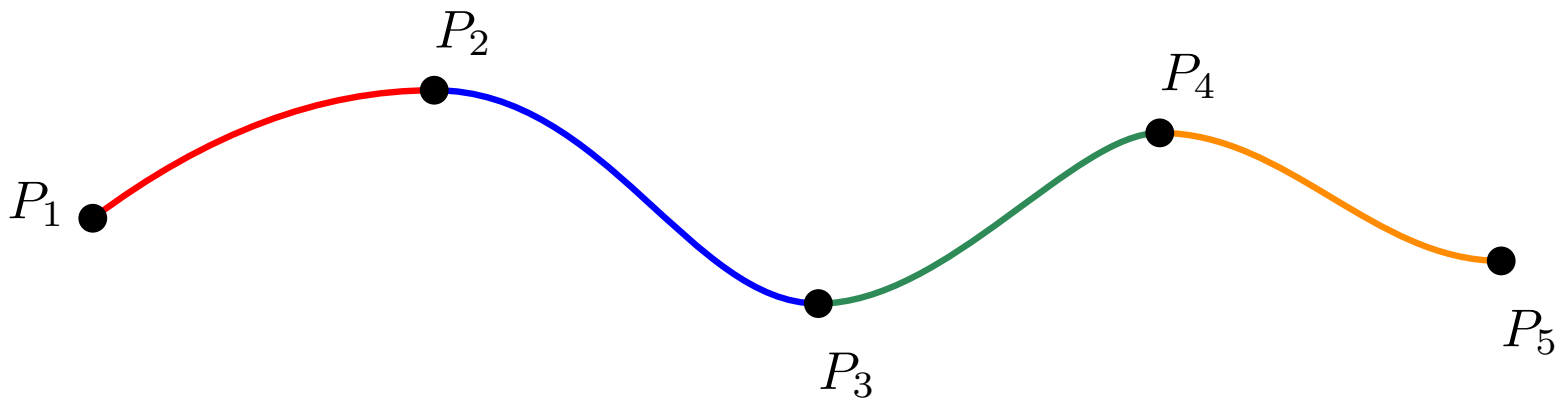
Example with $n = 5$



Cubic splines

- Input: n points $P_1, \ldots, P_n$ and 2 tangent vectors $\overrightarrow{v_1}$, $\overrightarrow{v_n}$ (other possibilities exist, see later)
- Output: A curve consisting of $n-1$ cubic Hermite curves that are smoothly connected (up to 2nd derivative) at the intermediate points $P_2, \ldots, P_{n-1}$
  this is referred to as being *C^2-continuous*

Example with $n = 5$



Computing cubic splines

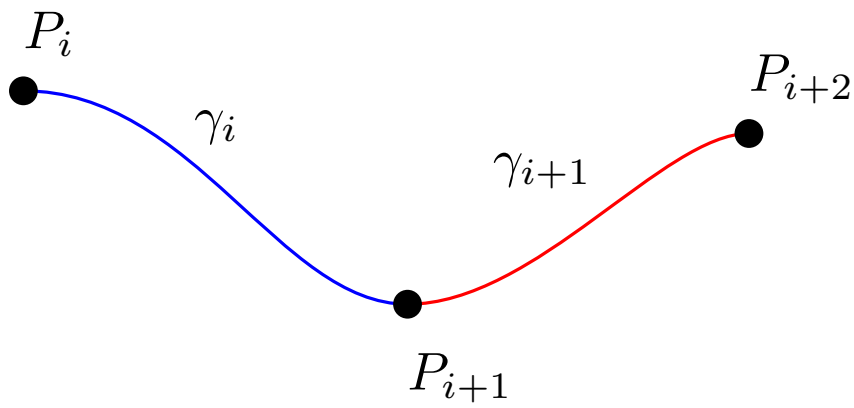
Consider three consecutive input points P_i, P_{i+1}, P_{i+2} for $i = 1, \ldots, n-2$.

Let γ_i be the Hermite curve connecting P_i to P_{i+1}, and γ_{i+1} the one connecting P_{i+1} to P_{i+2}

Computing cubic splines

Consider three consecutive input points P_i, P_{i+1}, P_{i+2} for $i = 1, \ldots, n-2$.

Let γ_i be the Hermite curve connecting P_i to P_{i+1}, and γ_{i+1} the one connecting P_{i+1} to P_{i+2}

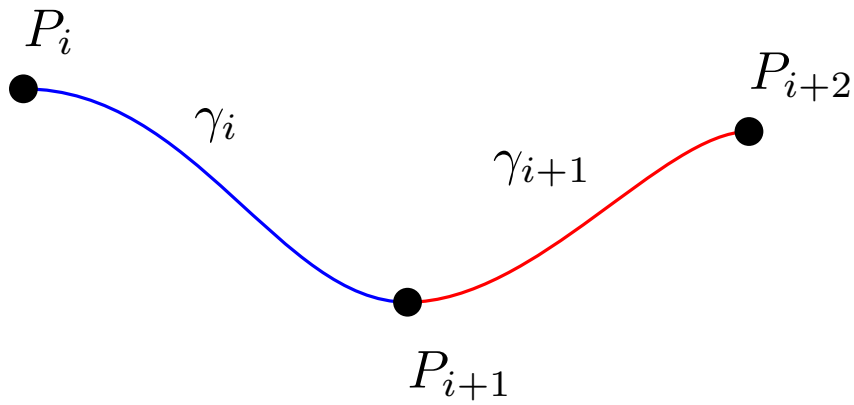


Computing cubic splines

Consider three consecutive input points P_i, P_{i+1}, P_{i+2} for $i = 1, \ldots, n-2$.

Let γ_i be the Hermite curve connecting P_i to P_{i+1}, and γ_{i+1} the one connecting P_{i+1} to P_{i+2}

Strategy: we will specify a system of equations with the conditions that our curves should satisfy, and solve the system to find out how they must look like

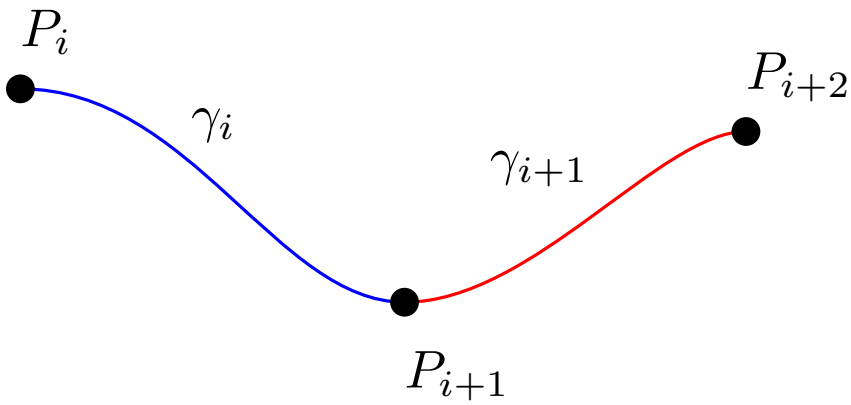


Computing cubic splines

Consider three consecutive input points P_i, P_{i+1}, P_{i+2} for $i = 1, \ldots, n-2$.

Let γ_i be the Hermite curve connecting P_i to P_{i+1}, and γ_{i+1} the one connecting P_{i+1} to P_{i+2}

Strategy: we will specify a system of equations with the conditions that our curves should satisfy, and solve the system to find out how they must look like



Curves γ_i and γ_{i+1} should satisfy the following:

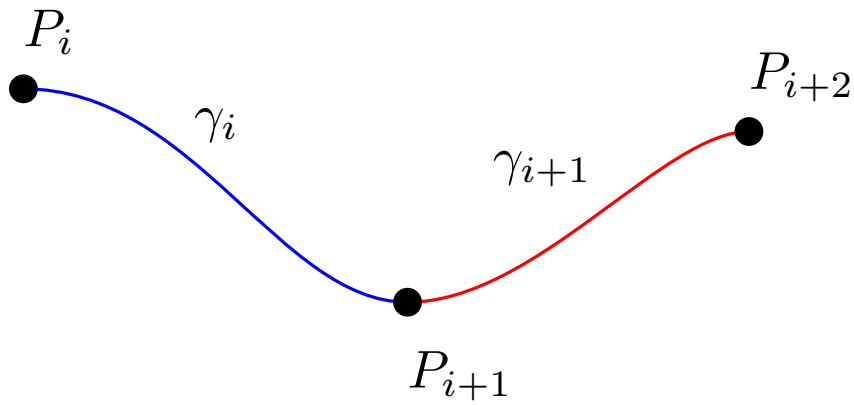
$$\gamma_i(0) = P_i$$
$$\left.\begin{aligned}\gamma_i(1) &= P_{i+1}\\ \gamma_{i+1}(0) &= P_{i+1}\end{aligned}\right\} =$$
$$\gamma_{i+1}(1) = P_{i+2}$$

Computing cubic splines

Consider three consecutive input points P_i, P_{i+1}, P_{i+2} for $i = 1, \ldots, n-2$.

Let γ_i be the Hermite curve connecting P_i to P_{i+1}, and γ_{i+1} the one connecting P_{i+1} to P_{i+2}

Strategy: we will specify a system of equations with the conditions that our curves should satisfy, and solve the system to find out how they must look like



Curves γ_i and γ_{i+1} should satisfy the following:

$$\left.\begin{aligned}\gamma_i(0) &= P_i\\ \gamma_i(1) &= P_{i+1}\\ \gamma_{i+1}(0) &= P_{i+1}\end{aligned}\right\}=$$

$$\left.\begin{aligned}\gamma_i'(0) &= \overrightarrow{v_i}\\ \gamma_i'(1) &= \overrightarrow{v_{i+1}}\\ \gamma_{i+1}'(0) &= \overrightarrow{v_{i+1}}\end{aligned}\right\}=$$

$$\gamma_{i+1}(1) = P_{i+2} \qquad \gamma_{i+1}'(1) = \overrightarrow{v_{i+2}}$$

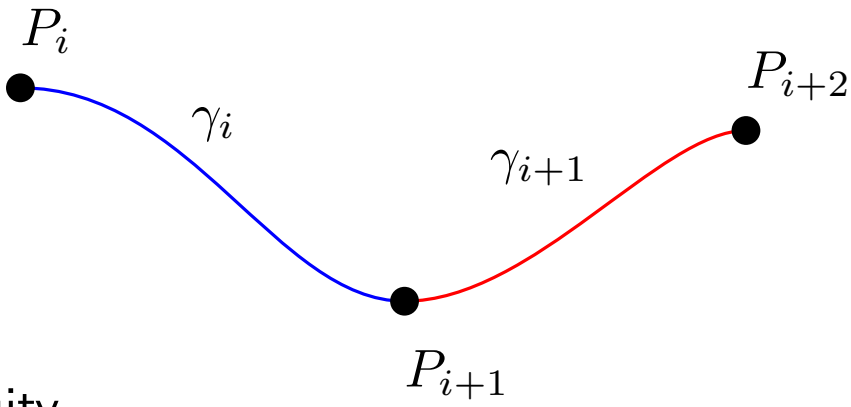
$$\gamma_i''(1) = \gamma_{i+1}''(0)$$

Computing cubic splines

Consider three consecutive input points P_i, P_{i+1}, P_{i+2} for $i = 1, \ldots, n-2$.

Let γ_i be the Hermite curve connecting P_i to P_{i+1}, and γ_{i+1} the one connecting P_{i+1} to P_{i+2}

Strategy: we will specify a system of equations with the conditions that our curves should satisfy, and solve the system to find out how they must look like



Curves γ_i and γ_{i+1} should satisfy the following:

$$\left.\begin{array}{l}\gamma_i(0) = P_i \\ \gamma_i(1) = P_{i+1} \\ \gamma_{i+1}(0) = P_{i+1} \\ \gamma_{i+1}(1) = P_{i+2}\end{array}\right\} C^0\text{-continuity}$$

$$\left.\begin{array}{l}\gamma_i'(0) = \overrightarrow{v_i} \\ \gamma_i'(1) = \overrightarrow{v_{i+1}} \\ \gamma_{i+1}'(0) = \overrightarrow{v_{i+1}} \\ \gamma_{i+1}'(1) = \overrightarrow{v_{i+2}}\end{array}\right\} C^1\text{-continuity}$$

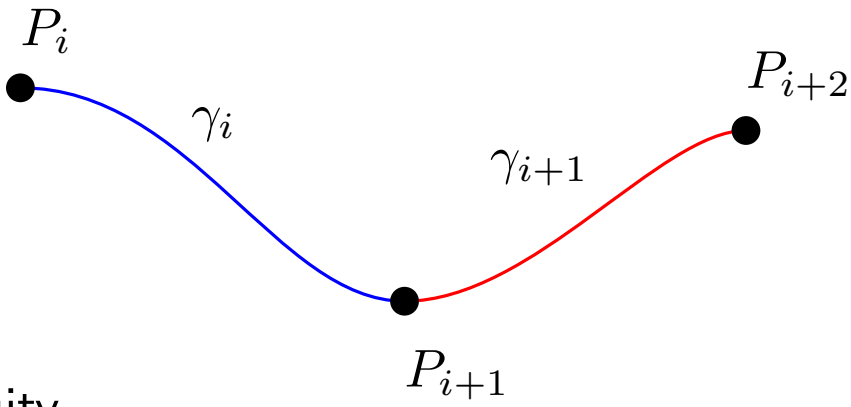
$$\left.\begin{array}{l}\gamma_i''(1) = \gamma_{i+1}''(0)\end{array}\right\} C^2\text{-continuity}$$

Computing cubic splines

Consider three consecutive input points P_i, P_{i+1}, P_{i+2} for $i = 1, \ldots, n-2$.

Let γ_i be the Hermite curve connecting P_i to P_{i+1}, and γ_{i+1} the one connecting P_{i+1} to P_{i+2}

Strategy: we will specify a system of equations with the conditions that our curves should satisfy, and solve the system to find out how they must look like



Curves γ_i and γ_{i+1} should satisfy the following:

$$\gamma_i(0) = P_i$$
$$\gamma_i(1) = P_{i+1} = \gamma_{i+1}(0)$$
$$\gamma_{i+1}(1) = P_{i+2}$$

known

C^0-continuity

$$\gamma_i'(0) = \overrightarrow{v_i}$$
$$\gamma_i'(1) = \overrightarrow{v_{i+1}} = \gamma_{i+1}'(0)$$
$$\gamma_{i+1}'(1) = \overrightarrow{v_{i+2}}$$

C^1-continuity

$$\gamma_i''(1) = \gamma_{i+1}''(0)$$

C^2-continuity

Computing cubic splines

Consider three consecutive input points P_i, P_{i+1}, P_{i+2} for $i = 1, \ldots, n-2$.

Let γ_i be the Hermite curve connecting P_i to P_{i+1}, and γ_{i+1} the one connecting P_{i+1} to P_{i+2}

Strategy: we will specify a system of equations with the conditions that our curves should satisfy, and solve the system to find out how they must look like

Curves γ_i and γ_{i+1} should satisfy the following:

$$\gamma_i(0) = P_i$$
$$\gamma_i(1) = P_{i+1} \quad \text{known}$$
$$\gamma_{i+1}(0) = P_{i+1} \quad =$$
$$\gamma_{i+1}(1) = P_{i+2}$$

C^0-continuity

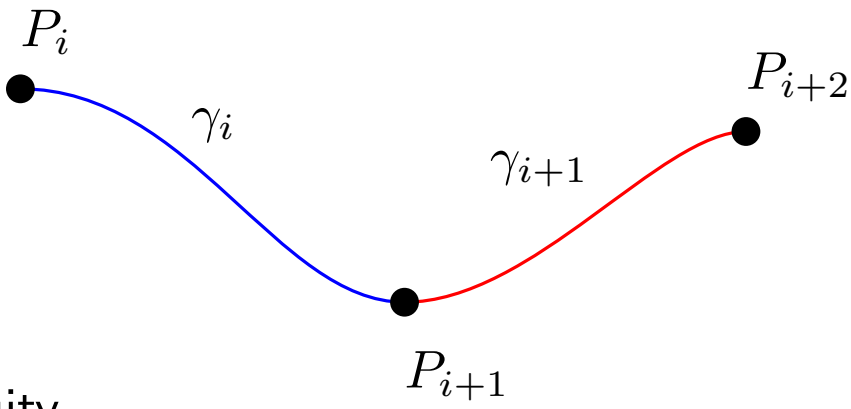
$$\gamma_i'(0) = \overrightarrow{v_i}$$
$$\gamma_i'(1) = \overrightarrow{v_{i+1}} \quad =$$
$$\gamma_{i+1}'(0) = \overrightarrow{v_{i+1}}$$
$$\gamma_{i+1}'(1) = \overrightarrow{v_{i+2}}$$

C^1-continuity

unknown!

$$\gamma_i''(1) = \gamma_{i+1}''(0)$$

C^2-continuity



$\rightarrow$ We don't know the tangent and second derivative vectors at the interior points! That is all that we need for the Hermite curves to be fully defined

Computing cubic splines

We begin from the equation of the Hermite curve γ_i and its first two derivatives

$$\gamma_i(t) = (2t^3 - 3t^2 + 1)P_i + (-2t^3 + 3t^2)P_{i+1} + (t^3 - 2t^2 + t)\overrightarrow{v_i} + (t^3 - t^2)\overrightarrow{v_{i+1}}$$

Computing cubic splines

We begin from the equation of the Hermite curve γ_i and its first two derivatives

$$\gamma_i(t) = (2t^3 - 3t^2 + 1)P_i + (-2t^3 + 3t^2)P_{i+1} + (t^3 - 2t^2 + t)\overrightarrow{v_i} + (t^3 - t^2)\overrightarrow{v_{i+1}}$$

$$\gamma_i'(t) = (6t^2 - 6t)P_i + (-6t^2 + 6t)P_{i+1} + (3t^2 - 4t + 1)\overrightarrow{v_i} + (3t^2 - 2t)\overrightarrow{v_{i+1}}$$

$$\gamma_i''(t) = (12t - 6)P_i + (-12t + 6)P_{i+1} + (6t - 4)\overrightarrow{v_i} + (6t - 2)\overrightarrow{v_{i+1}}$$

Computing cubic splines

We begin from the equation of the Hermite curve γ_i and its first two derivatives

$$\gamma_i(t) = (2t^3 - 3t^2 + 1)P_i + (-2t^3 + 3t^2)P_{i+1} + (t^3 - 2t^2 + t)\overrightarrow{v_i} + (t^3 - t^2)\overrightarrow{v_{i+1}}$$

$$\gamma_i'(t) = (6t^2 - 6t)P_i + (-6t^2 + 6t)P_{i+1} + (3t^2 - 4t + 1)\overrightarrow{v_i} + (3t^2 - 2t)\overrightarrow{v_{i+1}}$$

$$\gamma_i''(t) = (12t - 6)P_i + (-12t + 6)P_{i+1} + (6t - 4)\overrightarrow{v_i} + (6t - 2)\overrightarrow{v_{i+1}}$$

Therefore, using that $\gamma_i''(1) = \gamma_{i+1}''(0)$, we obtain:

Computing cubic splines

We begin from the equation of the Hermite curve γ_i and its first two derivatives

$$\gamma_i(t) = (2t^3 - 3t^2 + 1)P_i + (-2t^3 + 3t^2)P_{i+1} + (t^3 - 2t^2 + t)\overrightarrow{v_i} + (t^3 - t^2)\overrightarrow{v_{i+1}}$$

$$\gamma_i'(t) = (6t^2 - 6t)P_i + (-6t^2 + 6t)P_{i+1} + (3t^2 - 4t + 1)\overrightarrow{v_i} + (3t^2 - 2t)\overrightarrow{v_{i+1}}$$

$$\gamma_i''(t) = (12t - 6)P_i + (-12t + 6)P_{i+1} + (6t - 4)\overrightarrow{v_i} + (6t - 2)\overrightarrow{v_{i+1}}$$

Therefore, using that $\gamma_i''(1) = \gamma_{i+1}''(0)$, we obtain:

$$6P_i - 6P_{i+1} + 2\overrightarrow{v_i} + 4\overrightarrow{v_{i+1}} = -6P_{i+1} + 6P_{i+2} - 4\overrightarrow{v_{i+1}} - 2\overrightarrow{v_{i+2}}$$
$$\iff \overrightarrow{v_i} + 4\overrightarrow{v_{i+1}} + \overrightarrow{v_{i+2}} = 3(P_{i+2} - P_i)$$

Computing cubic splines

We begin from the equation of the Hermite curve γ_i and its first two derivatives

$$\gamma_i(t) = (2t^3 - 3t^2 + 1)P_i + (-2t^3 + 3t^2)P_{i+1} + (t^3 - 2t^2 + t)\overrightarrow{v_i} + (t^3 - t^2)\overrightarrow{v_{i+1}}$$

$$\gamma_i'(t) = (6t^2 - 6t)P_i + (-6t^2 + 6t)P_{i+1} + (3t^2 - 4t + 1)\overrightarrow{v_i} + (3t^2 - 2t)\overrightarrow{v_{i+1}}$$

$$\gamma_i''(t) = (12t - 6)P_i + (-12t + 6)P_{i+1} + (6t - 4)\overrightarrow{v_i} + (6t - 2)\overrightarrow{v_{i+1}}$$

Therefore, using that $\gamma_i''(1) = \gamma_{i+1}''(0)$, we obtain:

$$6P_i - 6P_{i+1} + 2\overrightarrow{v_i} + 4\overrightarrow{v_{i+1}} = -6P_{i+1} + 6P_{i+2} - 4\overrightarrow{v_{i+1}} - 2\overrightarrow{v_{i+2}}$$

$$\iff \overrightarrow{v_i} + 4\overrightarrow{v_{i+1}} + \overrightarrow{v_{i+2}} = 3(P_{i+2} - P_i) \qquad \text{for } i = 1, \ldots, n-2$$

unknown known

Computing cubic splines

We begin from the equation of the Hermite curve γ_i and its first two derivatives

$$\gamma_i(t) = (2t^3 - 3t^2 + 1)P_i + (-2t^3 + 3t^2)P_{i+1} + (t^3 - 2t^2 + t)\overrightarrow{v_i} + (t^3 - t^2)\overrightarrow{v_{i+1}}$$

$$\gamma_i'(t) = (6t^2 - 6t)P_i + (-6t^2 + 6t)P_{i+1} + (3t^2 - 4t + 1)\overrightarrow{v_i} + (3t^2 - 2t)\overrightarrow{v_{i+1}}$$

$$\gamma_i''(t) = (12t - 6)P_i + (-12t + 6)P_{i+1} + (6t - 4)\overrightarrow{v_i} + (6t - 2)\overrightarrow{v_{i+1}}$$

Therefore, using that $\gamma_i''(1) = \gamma_{i+1}''(0)$, we obtain:

$$6P_i - 6P_{i+1} + 2\overrightarrow{v_i} + 4\overrightarrow{v_{i+1}} = -6P_{i+1} + 6P_{i+2} - 4\overrightarrow{v_{i+1}} - 2\overrightarrow{v_{i+2}}$$

$$\iff \overrightarrow{v_i} + 4\overrightarrow{v_{i+1}} + \overrightarrow{v_{i+2}} = 3(P_{i+2} - P_i) \qquad \text{for } i = 1, \ldots, n-2$$

unknown known

We have $n-2$ equations like this. That gives a system of $n-2$ equations where the variables are the $n-2$ unknown tangent vectors $\overrightarrow{v_2}, \ldots, \overrightarrow{v_{n-1}}$ (recall that $\overrightarrow{v_1}, \overrightarrow{v_n}$ are given)

Computing cubic splines

Each equation has shape $\overrightarrow{v_i} + 4\overrightarrow{v_{i+1}} + \overrightarrow{v_{i+2}} = 3(P_{i+2} - P_i)$ for $i = 1, \ldots, n-2$

We can write the equation system in matrix form as follows:

Computing cubic splines

Each equation has shape $\overrightarrow{v_i} + 4\overrightarrow{v_{i+1}} + \overrightarrow{v_{i+2}} = 3(P_{i+2} - P_i)$ for $i = 1, \ldots, n-2$

We can write the equation system in matrix form as follows:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & \ldots & 0 & 0 & 0 & 0 \\ 1 & 4 & 1 & 0 & \ldots & 0 & 0 & 0 & 0 \\ 0 & 1 & 4 & 1 & \ldots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \ldots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \ldots & 0 & 1 & 4 & 1 \\ 0 & 0 & 0 & 0 & \ldots & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \overrightarrow{v_1} \\ \overrightarrow{v_2} \\ \overrightarrow{v_3} \\ \vdots \\ \overrightarrow{v_{n-1}} \\ \overrightarrow{v_n} \end{pmatrix} = \begin{pmatrix} \text{user value} \\ 3(P_3 - P_1) \\ 3(P_4 - P_2) \\ \vdots \\ 3(P_n - P_{n-2}) \\ \text{user value} \end{pmatrix}$$

Computing cubic splines

Each equation has shape $\overrightarrow{v_i} + 4\overrightarrow{v_{i+1}} + \overrightarrow{v_{i+2}} = 3(P_{i+2} - P_i)$ for $i = 1, \ldots, n-2$

We can write the equation system in matrix form as follows:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & \ldots & 0 & 0 & 0 & 0 \\ 1 & 4 & 1 & 0 & \ldots & 0 & 0 & 0 & 0 \\ 0 & 1 & 4 & 1 & \ldots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \ldots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \ldots & 0 & 1 & 4 & 1 \\ 0 & 0 & 0 & 0 & \ldots & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \overrightarrow{v_1} \\ \overrightarrow{v_2} \\ \overrightarrow{v_3} \\ \vdots \\ \overrightarrow{v_{n-1}} \\ \overrightarrow{v_n} \end{pmatrix} = \begin{pmatrix} \underline{\text{user value}} \\ 3(P_3 - P_1) \\ 3(P_4 - P_2) \\ \vdots \\ 3(P_n - P_{n-2}) \\ \underline{\text{user value}} \end{pmatrix}$$

Note that this includes the two input tangent vectors, thus it has n equations and n variables

Computing cubic splines

Each equation has shape $\overrightarrow{v_i} + 4\overrightarrow{v_{i+1}} + \overrightarrow{v_{i+2}} = 3(P_{i+2} - P_i)$ for $i = 1, \ldots, n-2$

We can write the equation system in matrix form as follows:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & \ldots & 0 & 0 & 0 & 0 \\ 1 & 4 & 1 & 0 & \ldots & 0 & 0 & 0 & 0 \\ 0 & 1 & 4 & 1 & \ldots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \ldots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \ldots & 0 & 1 & 4 & 1 \\ 0 & 0 & 0 & 0 & \ldots & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \overrightarrow{v_1} \\ \overrightarrow{v_2} \\ \overrightarrow{v_3} \\ \vdots \\ \overrightarrow{v_{n-1}} \\ \overrightarrow{v_n} \end{pmatrix} = \begin{pmatrix} \underline{\text{user value}} \\ 3(P_3 - P_1) \\ 3(P_4 - P_2) \\ \vdots \\ 3(P_n - P_{n-2}) \\ \underline{\text{user value}} \end{pmatrix}$$

Note that this includes the two input tangent vectors, thus it has n equations and n variables

This system has rank n, so it has a unique solution!

Summary: to produce the cubic splines one has to:
(i) Solve the system to obtain the $n-2$ unknown tangent vectors
(ii) Use them to build the $n-1$ Hermite curves connecting each pair of consecutive points P_i, P_{i+1}, for $i = 1, \ldots, n-1$

Variations of cubic splines

The version we just saw, where the user specifies, in addition to the data points, the first and last tangent vectors, is known as *clamped* cubic spline. Other variants exist. For example:

Variations of cubic splines

The version we just saw, where the user specifies, in addition to the data points, the first and last tangent vectors, is known as *clamped* cubic spline. Other variants exist. For example:

- Relaxed cubic splines

Instead of specifying $\overrightarrow{v_1}, \overrightarrow{v_n}$, specify $\gamma'' = 0$ at the two endpoints of the curve: $\gamma_1''(0) = 0$ and $\gamma_{n-1}''(1) = 0$. This has the effect of producing low curvature at the ends of the curve.

Variations of cubic splines

The version we just saw, where the user specifies, in addition to the data points, the first and last tangent vectors, is known as *clamped* cubic spline. Other variants exist. For example:

- Relaxed cubic splines

Instead of specifying $\overrightarrow{v_1}, \overrightarrow{v_n}$, specify $\gamma'' = 0$ at the two endpoints of the curve: $\gamma_1''(0) = 0$ and $\gamma_{n-1}''(1) = 0$. This has the effect of producing low curvature at the ends of the curve.

- Cyclic cubic splines

Instead of specifying $\overrightarrow{v_1}, \overrightarrow{v_n}$, specify that the tangent and second derivative are the same at the endpoints: $\gamma_0'(0) = \gamma_{n-1}'(1)$ and $\gamma_0''(0) = \gamma_{n-1}''(1)$. This is often used for curves that are closed or periodic.

Variations of cubic splines

The version we just saw, where the user specifies, in addition to the data points, the first and last tangent vectors, is known as *clamped* cubic spline. Other variants exist. For example:

- Relaxed cubic splines

Instead of specifying $\overrightarrow{v_1}, \overrightarrow{v_n}$, specify $\gamma'' = 0$ at the two endpoints of the curve: $\gamma_1''(0) = 0$ and $\gamma_{n-1}''(1) = 0$. This has the effect of producing low curvature at the ends of the curve.

- Cyclic cubic splines

Instead of specifying $\overrightarrow{v_1}, \overrightarrow{v_n}$, specify that the tangent and second derivative are the same at the endpoints: $\gamma_0'(0) = \gamma_{n-1}'(1)$ and $\gamma_0''(0) = \gamma_{n-1}''(1)$. This is often used for curves that are closed or periodic.

- Closed cubic splines

It has an extra curve γ_n from P_n to P_1. It can be implemented by adding two dummy points P_{n+1}, P_{n+2}, defined as $P_{n+1} = P_1$ and $P_{n+2} = P_2$

Variations of cubic splines

The version we just saw, where the user specifies, in addition to the data points, the first and last tangent vectors, is known as *clamped* cubic spline. Other variants exist. For example:

- Relaxed cubic splines

Instead of specifying $\overrightarrow{v_1}, \overrightarrow{v_n}$, specify $\gamma'' = 0$ at the two endpoints of the curve: $\gamma_1''(0) = 0$ and $\gamma_{n-1}''(1) = 0$. This has the effect of producing low curvature at the ends of the curve.

- Cyclic cubic splines

Instead of specifying $\overrightarrow{v_1}, \overrightarrow{v_n}$, specify that the tangent and second derivative are the same at the endpoints: $\gamma_0'(0) = \gamma_{n-1}'(1)$ and $\gamma_0''(0) = \gamma_{n-1}''(1)$. This is often used for curves that are closed or periodic.

- Closed cubic splines

It has an extra curve γ_n from P_n to P_1. It can be implemented by adding two dummy points P_{n+1}, P_{n+2}, defined as $P_{n+1} = P_1$ and $P_{n+2} = P_2$

See [Salomon], sections 5.1.2, 5.1.3, 5.1.5 for more details

Issues with cubic splines

Splines have a few drawbacks. The two most important ones are:

Issues with cubic splines

Splines have a few drawbacks. The two most important ones are:

- Requires solving a system of equations of size $n \times n$, which can be large if n is large

Issues with cubic splines

Splines have a few drawbacks. The two most important ones are:

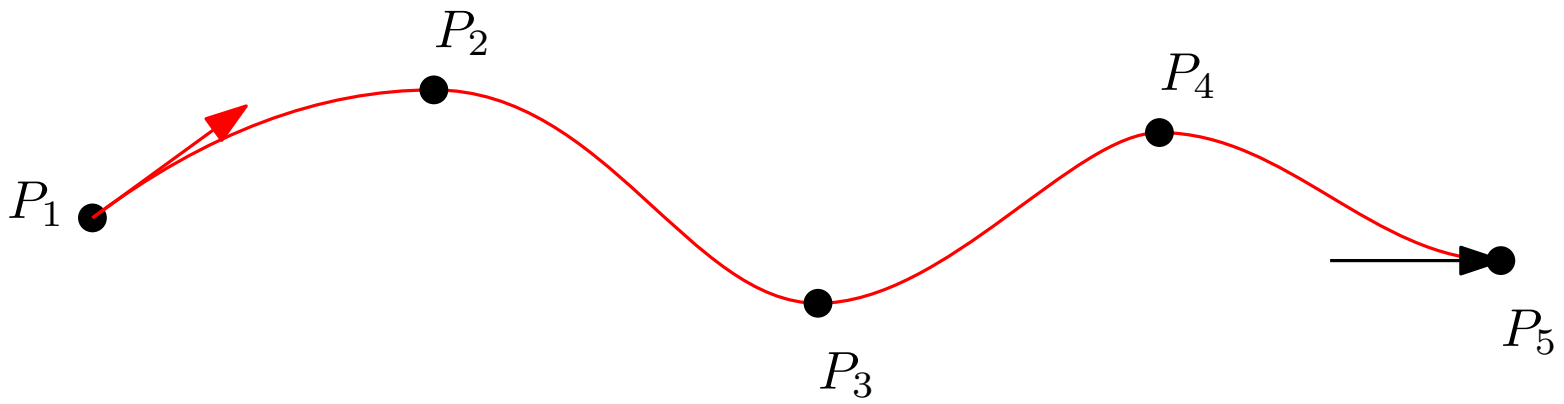
- Requires solving a system of equations of size $n \times n$, which can be large if n is large
- Splines lack *local control*: modifying one point or even one of the two end tangent vectors modifies the whole curve (and that is not good!)

Issues with cubic splines

Splines have a few drawbacks. The two most important ones are:

- Requires solving a system of equations of size $n \times n$, which can be large if n is large
- Splines lack *local control*: modifying one point or even one of the two end tangent vectors modifies the whole curve (and that is not good!)

Example: effect of modifying the initial tangent vector

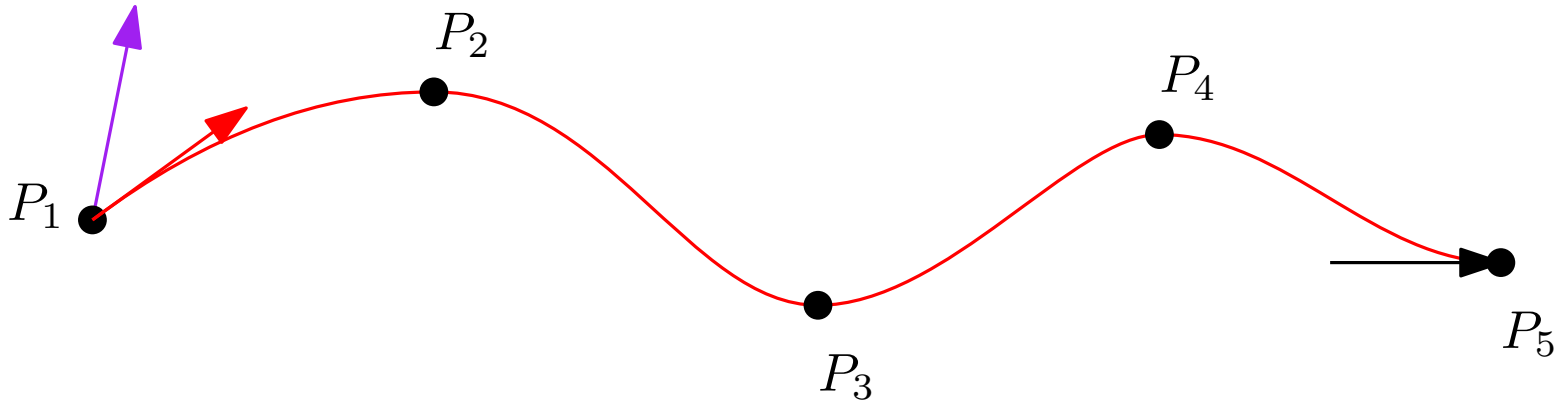


Issues with cubic splines

Splines have a few drawbacks. The two most important ones are:

- Requires solving a system of equations of size $n \times n$, which can be large if n is large
- Splines lack *local control*: modifying one point or even one of the two end tangent vectors modifies the whole curve (and that is not good!)

Example: effect of modifying the initial tangent vector

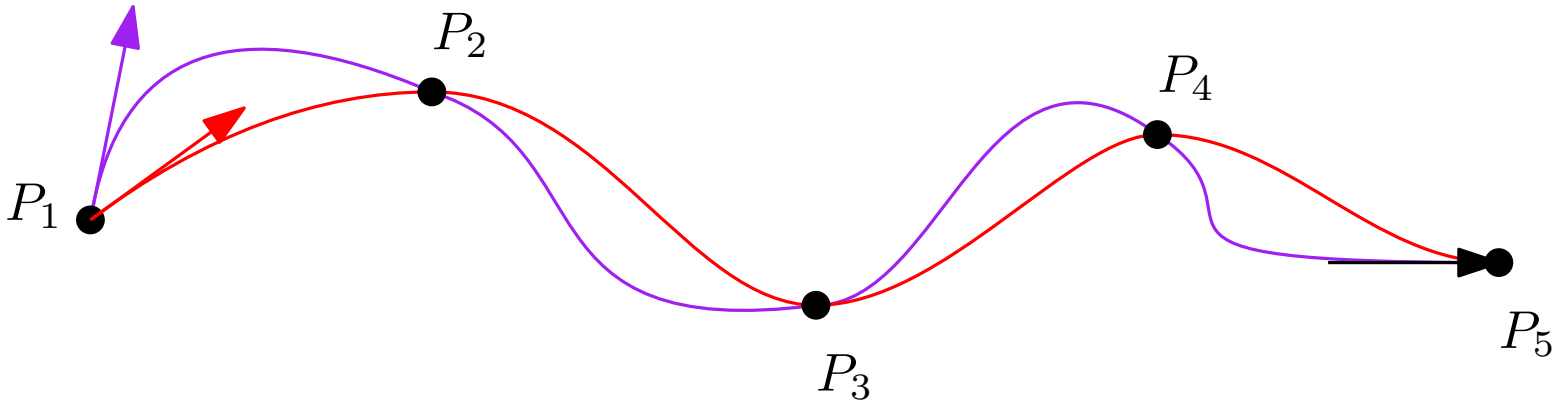


Issues with cubic splines

Splines have a few drawbacks. The two most important ones are:

- Requires solving a system of equations of size $n \times n$, which can be large if n is large
- Splines lack *local control*: modifying one point or even one of the two end tangent vectors modifies the whole curve (and that is not good!)

Example: effect of modifying the initial tangent vector



(the example is only approximate)

Cardinal splines: a way to overcome these limitations

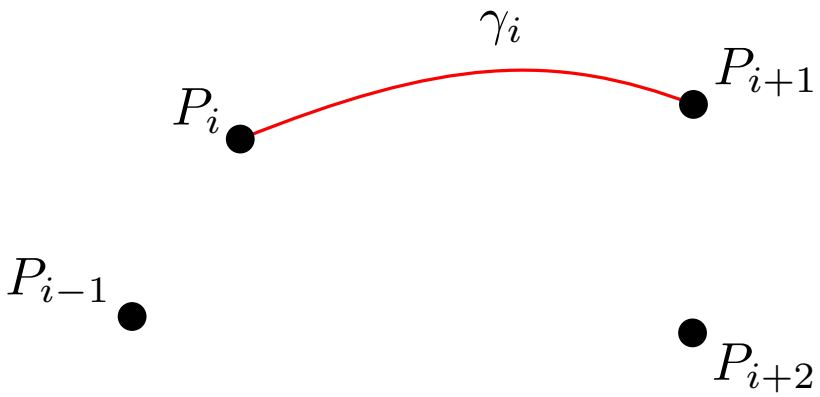
A different way to apply Hermite interpolation to construct a spline

- Each curve is defined based on only four consecutive data points
- Therefore, it has local control
- It is C^1-continuous, but not C^2-continuous (that's the price to pay!)

Cardinal splines: a way to overcome these limitations

A different way to apply Hermite interpolation to construct a spline

- Each curve is defined based on only four consecutive data points
- Therefore, it has local control
- It is C^1-continuous, but not C^2-continuous (that's the price to pay!)



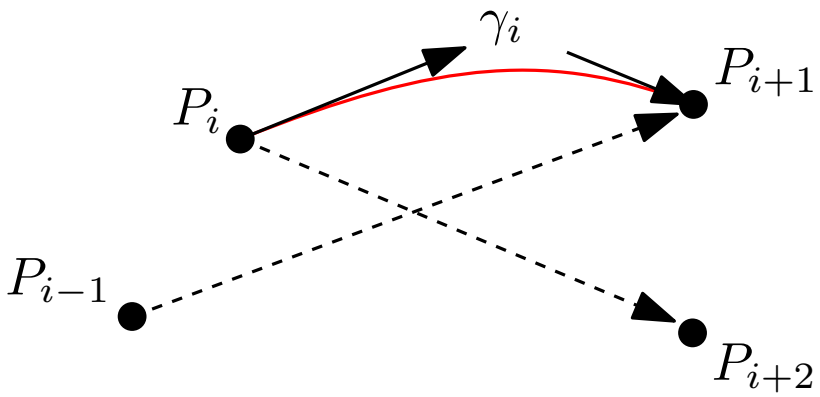
The Hermite curve γ_i from P_i to P_{i+1} is defined by four points: $P_{i-1}, P_i, P_{i+1}, P_{i+2}$

- it goes from P_i to P_{i+1}

Cardinal splines: a way to overcome these limitations

A different way to apply Hermite interpolation to construct a spline

- Each curve is defined based on only four consecutive data points
- Therefore, it has local control
- It is C^1-continuous, but not C^2-continuous (that's the price to pay!)



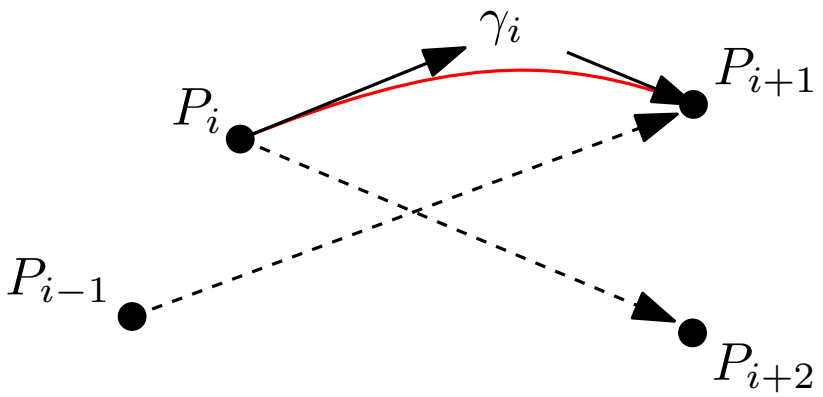
The Hermite curve γ_i from P_i to P_{i+1} is defined by four points: $P_{i-1}, P_i, P_{i+1}, P_{i+2}$

- it goes from P_i to P_{i+1}
- the initial tangent vector is given by $s(P_{i+1} - P_{i-1})$
- the end tangent vector is given by $s(P_{i+2} - P_i)$

Cardinal splines: a way to overcome these limitations

A different way to apply Hermite interpolation to construct a spline

- Each curve is defined based on only four consecutive data points
- Therefore, it has local control
- It is C^1-continuous, but not C^2-continuous (that's the price to pay!)



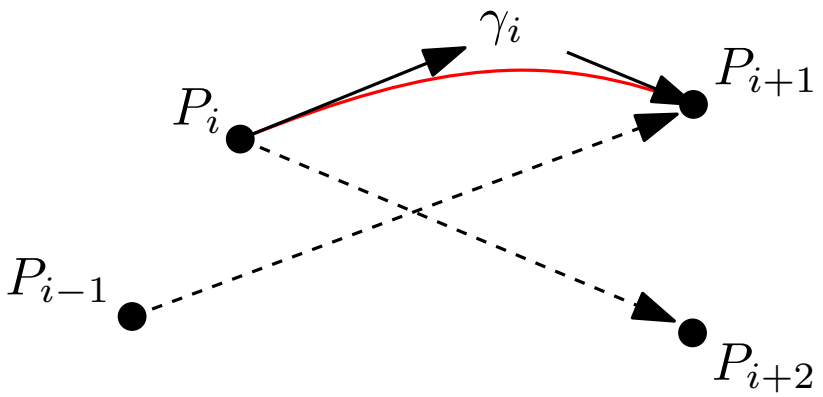
The Hermite curve γ_i from P_i to P_{i+1} is defined by four points: $P_{i-1}, P_i, P_{i+1}, P_{i+2}$

- it goes from P_i to P_{i+1}
- the initial tangent vector is given by $s(P_{i+1} - P_{i-1})$
- the end tangent vector is given by $s(P_{i+2} - P_i)$
- where $s \geq 0$ is a parameter that controls the tension of the curve (usually $s \in [0, 1]$)
- P_0 and P_{n+1} need to be defined by the user

Cardinal splines: a way to overcome these limitations

A different way to apply Hermite interpolation to construct a spline

- Each curve is defined based on only four consecutive data points
- Therefore, it has local control
- It is C^1-continuous, but not C^2-continuous (that's the price to pay!)



The Hermite curve γ_i from P_i to P_{i+1} is defined by four points: $P_{i-1}, P_i, P_{i+1}, P_{i+2}$

- it goes from P_i to P_{i+1}
- the initial tangent vector is given by $s(P_{i+1} - P_{i-1})$
- the end tangent vector is given by $s(P_{i+2} - P_i)$
- where $s \geq 0$ is a parameter that controls the tension of the curve (usually $s \in [0, 1]$)
- P_0 and P_{n+1} need to be defined by the user

Note that each Hermite curve depends only on four points, avoiding the $n \times n$ linear system

THIS CONCLUDES INTERPOLATION

This concludes our study of *interpolation* methods. Next, we will see that one doesn't need to interpolate to produce nice curves in an interactive way. The remaining of the course will focus on *approximation* methods.

Most of this and two previous presentations follow [Salomon], Section 3.2 (Lagrange interpolation) and Section 5 (Hermite interpolation)