S

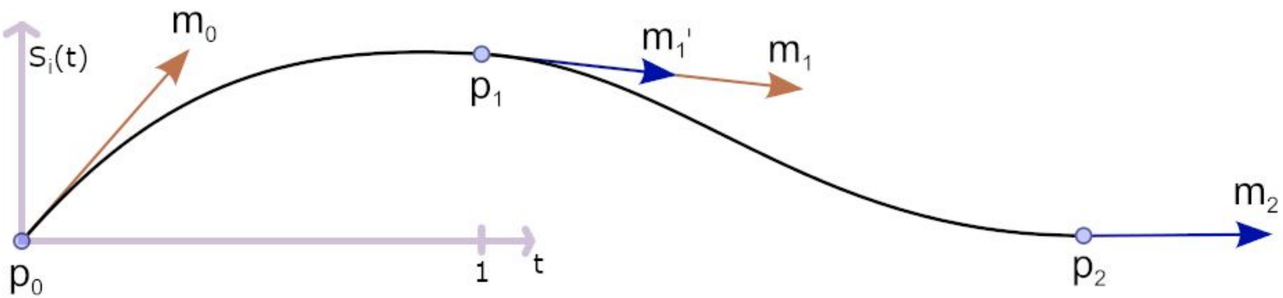
Splines

Moritz Marquardt & Shaikh Rezwan

What are Splines?

- Basic problem: we have a set of points - e. g. (0,0), (1,1) and (2,0) - and want to get a smooth curve instead of just two lines.
- Splines are piecewise polynomial parametric curves:
    - piecewise: we have different parts, separated by the points in our path
    - polynomial: each part can be represented by a polynomial function
    - parametric: the form of the curve can be manipulated using parameters

➜ $S_i(t_i)$ is the polynomial of the piece *i* dependent on t_i (between 0 and 1)



Cubic Hermite Spline [10]

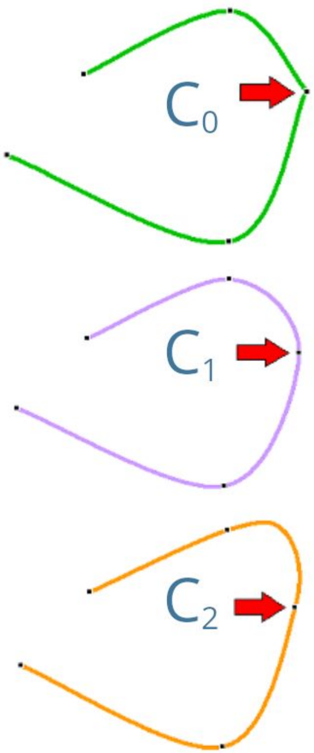
Why do we use Splines in Robotics?

- Without Splines: we can chain multiple LIN & PTP movements
  - abrupt acceleration (high jerk-rapid change in acceleration)
    - Jerk causes the system to vibrate,
    - vibrations decrease positioning accuracy,
    - while increasing settling time
  - de-/acceleration costs energy and time
- More flexibility than with just LIN & PTP movements
- We want our TCP to follow a smooth path while...
  - exactly hitting the given points
  - not having to provide too much additional parameters
  - being able to change points without affecting the rest of the path too much

S

Criteria for Splines

- Continuity: Splines can be continuous in...
  - position (C_0)
  - position & tangent vector (C_1, velocity is continuous)
  - position, tangent vector & curvature (C_2, acceleration is continuous)
- Strategy:
  - interpolation: the spline polynomial directly visits the specified points
  - approximation: polynomial only runs somewhere near those points
  - visiting points is required in a robotics context
- Local Control / Localism:
  - changes only affects the curve next to the changed points
  - probably makes it easier to use in a robotics context
- Required parameters
  - we need a little control of the movement path between the points
  - we also don’t want to set like 5 different parameters for each point



Continuity [2]

Applying Splines to Robotics

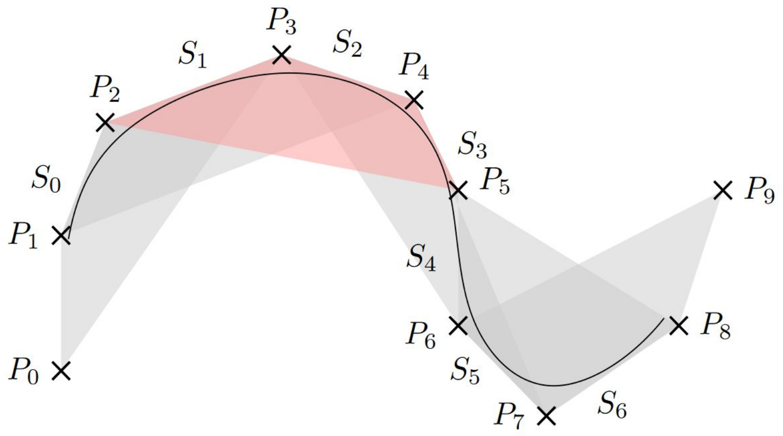
- For a continuous velocity, C_1 continuity of spline is good enough.
- C_2 is not totally a requirement, but mitigates vibrations & reduce wear on parts
- Interpolation is the required strategy for intuitive paths
- Ease of use is probably the most important aspect, so Catmull-Rom would make a lot of sense.
  ➔ Problem: the path won’t be controllable as well in some circumstances.
- Idea: Use Catmull-Rom for basic paths, with the possibility to use Bezier splines if required (maybe even just for a few segments)
- Splines are the most high-level interface & determine the tool path
  - Input is a set of points where the TCP should move
  - Output is a function of the position over time, which is then used for the trajectories

Types of Splines

Spline Type	Continuity	Strategy	Localism
B-Splines & NURBS	C_2	*approximation*	yes
Natural Cubic Splines	C_2	interpolation	*no*
Cubic Bezier	C_1	interpolation	yes
Quintic Bezier	$\mathbf{C_2}$	**interpolation**	**yes**
Cubic Hermite	C_1	interpolation	yes
Cubic Catmull-Rom	C_1	interpolation	yes

B-Splines

- A spline function that has minimal support with respect to a given degree, smoothness, and domain partition
- Any spline function of given degree can be expressed as a linear combination of B-splines of that degree.
- Cardinal B-splines have knots that are equidistant from each other.
- B-splines can be used for curve-fitting and numerical differentiation of experimental data.



B-Spline [3]

$$S(t) = \sum_{i=0}^{n} N_{i,k}(t) P_i,$$

$$N_{i,0}(t) = \begin{cases} 1 & \text{if } t_i \le t < t_{i+1} \\ 0 & \text{otherwise} \end{cases},$$

$$N_{i,j}(t) = \frac{t - t_i}{t_{i+j} - t_i} N_{i,j-1}(t) + \frac{t_{i+j+1} - t}{t_{i+j+1} - t_{i+1}} N_{i+1,j-1}(t).$$

NURBS: “Non-Uniform Rational B-Spline”

- non-uniform (parameters can have a physical meaning)
- rational (control points are weighted)
- easy to use, especially for surfaces

S

Natural Cubic Splines

- Generic view on cubic splines: find a solution for a third-degree polynomial between two points
- We need these conditions to fix the coefficients, plus 2 more:

(1) $s_i(x_i) = y_i,$ for $i = 0 : m-1,$

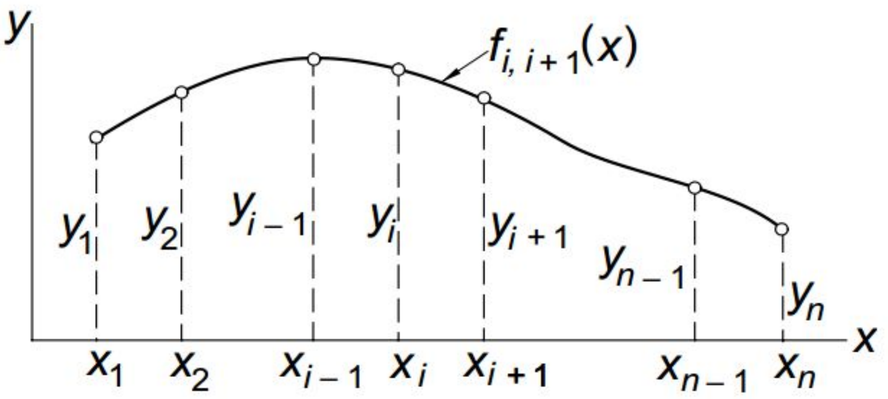
(2) $s_{m-1} = y_m,$ 1 condition

(3) $s_i(x_{i+1}) = s_{i+1}(x_{i+1}),$ for $i = 0 : m-2,$

(4) $s'_i(x_{i+1}) = s'_{i+1}(x_{i+1}),$ for $i = 0 : m-2,$

(5) $s''_i(x_{i+1}) = s''_{i+1}(x_{i+1}),$ for $i = 0 : m-2,$

Natural Cubic Splines [6]



$$s_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i$$

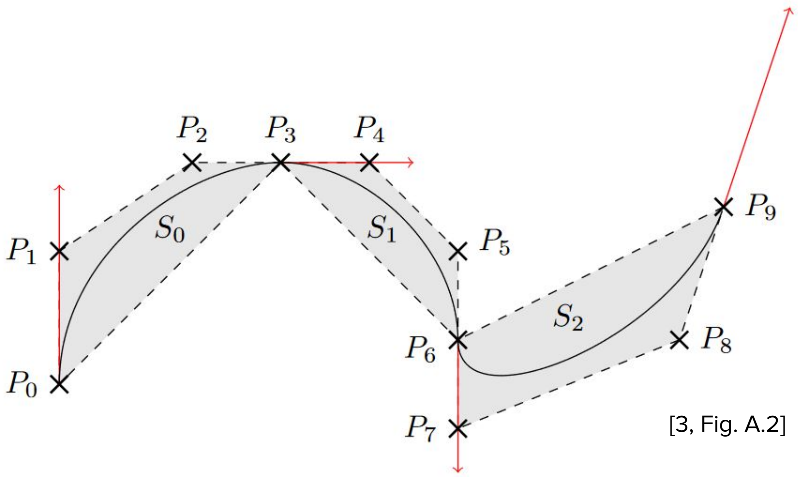
a. Natural Cubic Spline: $s''_0(x_0) = 0$ and $s''_{m-1}(x_m) = 0$ → C_2 continuity!

b. Not-a-knot Spline: $s'''_0(x_1) = s'''_1(x_1)$ and $s'''_{m-2}(x_{m-1}) = s'''_{m-1}(x_{m-1})$

- Calculation is a matter of solving a linear equation system
- Natural or generic cubic Splines are not very intuitive or flexible

Cubic Bezier Splines

- Each segment has 4 control points:
  - two of them will be visited (P0 & P3)
  - the other two provide directional information



[3, Fig. A.2]

- Each segment can be calculated like this:

$$B(t) = (1-t)^3 P_0 + (3-t)^2 P_1 + (3-t)^2 P_2 + t^3 P_3 \text{ with } 0 \leq t \leq 1$$

- How to achieve C_0 continuity?

  The line from the third control point of one segment to the second one of the next segment must subtend the last/first control point, and neither of them must be equal to that control point.

- How to achieve C_1 continuity?

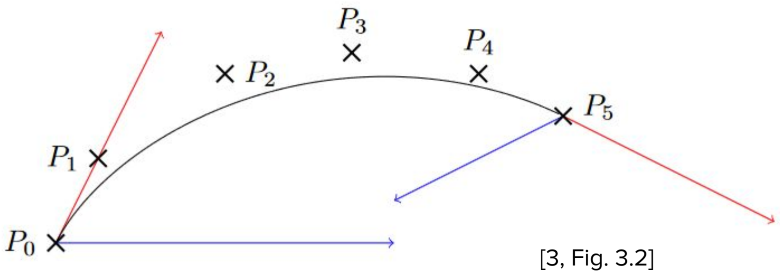
  The distance from the third control point of one segment to the second one of the next segment must subtend the last/first control point.

Quintic Bezier Splines

$$S(t) = (1-t)^5 P_0 + 5(1-t)^4 t P_1 + 10(1-t)^3 t^2 P_2 + 10(1-t)^2 t^3 P_3 + 5(1-t) t^4 P_4 + t^5 P_5$$

By using the right values for P_2 & P_3, Quintic Bezier Splines can be C_2 continuous.

Problem: 6 control points required - how to calculate them all?



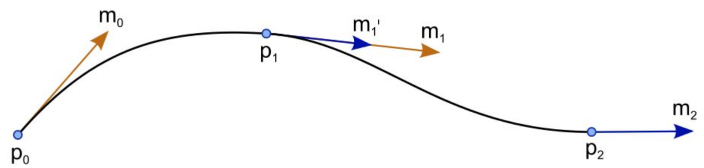
[3, Fig. 3.2]

By setting t and a to the wanted values of the first and second derivative, those control points can be calculated relatively easily:

$$P_1 = \frac{1}{5} t_s + P_0, \qquad P_4 = P_5 - \frac{1}{5} t_e$$

$$P_2 = \frac{1}{20} a_s + 2P_1 - P_0, \qquad P_3 = \frac{1}{20} a_e + 2P_4 - P_5.$$

Cubic Hermite Spline



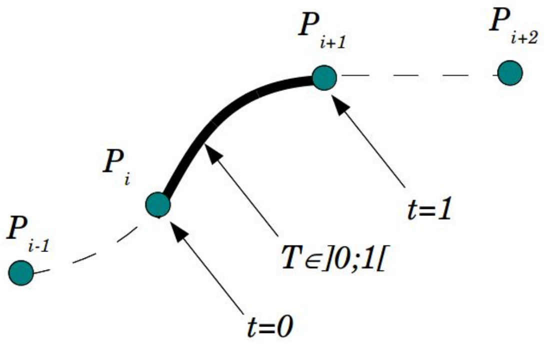
- Cubic Hermite spline: each piece is a third-degree polynomial specified by its values and first derivatives at the end points of the domain interval.
- Used for interpolation of numeric data specified at given values (e.g. x_1, x_2, ... , x_n) to obtain a continuous function.
- Requirements:
  - Desired function value
  - Derivative at each x_k
- The resulting spline will be C_1 continuous
- Problem: Must explicitly specify unintuitive derivatives at each endpoint!
- Equation: $p(t) = (2t^3 - 3t^2 + 1)p_0 + (t^3 - 2t^2 + t)m_0 + (-2t^3 + 3t^2)p_1 + (t^3 - t^2)m_1$

M

Catmull-Rom Spline

- The problem of the Cubic Hermite spline can be solved using Catmull-Rom spline. It has built-in C_1 continuity.
- The tangent at each point p_i is calculated using the previous and next point on the spline, $\mathscr{T}(p_{i+1} - p_{i-1})$, where, the parameter $\mathscr{T}$ is known as “tension”, and it affects how sharply the curve bends at the interpolated control points.
- Advantages:
  - It will not form loop or self-intersection within a curve segment
  - Cusp will never occur within a curve segment
  - Follows the control points tightly

$$P = \begin{bmatrix} 1 & t & t^2 & t^3 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\tau & 0 & \tau & 0 \\ 2.\tau & \tau - 3 & 3 - 2\tau & -\tau \\ -\tau & 2 - \tau & \tau - 2 & \tau \end{bmatrix} \cdot \begin{bmatrix} P_{i-1} & P_i & P_{i+1} & P_{i+2} \end{bmatrix}^\top$$



Catmull-Rom Spline Interpolation with 4 points [8]

M

Splines in Three Dimensions

- We will get a set of points in a three-dimensional space, and are expected to return a function which returns a point in that space dependent on *t*.
- **e. g. Bezier Splines: the same equation can be used for any dimension!**

- To use other spline functions which don't work that nicely with multi-dimensional points, we can just use three piecewise polynomials $S_x(t)$, $S_y(t)$ and $S_z(t)$, with the resulting path being the following function:

$$S(t) = \begin{pmatrix} S_x(t) \\ S_y(t) \\ S_z(t) \end{pmatrix}$$

M

How to use Splines for multiple points?

Input: $P_1, P_2, ..., P_n; 0 \leq t \leq 1$

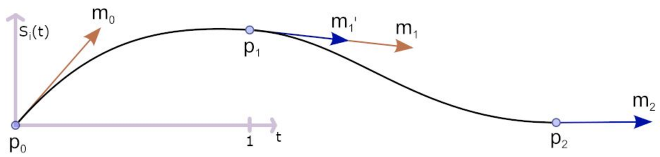
Calculate distance between points to get the bounds for t_n for each piece:

$$d(i) = \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2 + (z_i - z_{i+1})^2}$$

$$b(i) \leq t_i \leq b(i+1) \text{ with } b(i) = \sum_{j=1}^{i-1} d(j)$$

Create the piecewise polynomial from the different parts:

$$S(t) = \begin{cases} S_i(\frac{t-b(i)}{d(i)}) & b(i-1) \leq t < b(i); \text{ with } i = 1, 2, ..., n \\ S_n(t) & b(n) \leq t \end{cases}$$



To improve performance, ensure that *b(i)* and *d(i)* are only calculated once.

TL;DR

- Splines are used to create a smooth tool path
- Use Quintic Bezier Splines for a C_2 continuous implementation:

$$S(t) = (1-t)^5 P_0 + 5(1-t)^4 t P_1 + 10(1-t)^3 t^2 P_2 + 10(1-t)^2 t^3 P_3 + 5(1-t) t^4 P_4 + t^5 P_5$$

- Use t & a to calculate missing points:

$$P_1 = \frac{1}{5} t_s + P_0, \qquad P_4 = P_5 - \frac{1}{5} t_e$$

$$P_2 = \frac{1}{20} a_s + 2P_1 - P_0, \qquad P_3 = \frac{1}{20} a_e + 2P_4 - P_5.$$

- See the previous slide for stitching of multiple segments
- Use constant orientation for the implementation (only X/Y/Z for Splines)

M

Resources

[1] Types of Splines (Andrew Marriott, Curtin University)
http://euklid.mi.uni-koeln.de/c/mirror/www.cs.curtin.edu.au/units/cg351-551/notes/lect6c1.html

[2] Computer Graphics - Splines (Jessica K. Hodgins, Carnegie Mellon University)
http://www.cs.cmu.edu/~jkh/462_s07/07_curves_splines_part1.pdf
http://www.cs.cmu.edu/~jkh/462_s07/08_curves_splines_part2.pdf

[3] Planning Motion Trajectories for Mobile Robots Using Splines (Christoph Sprunk, Universität Freiburg)
http://www2.informatik.uni-freiburg.de/~lau/students/Sprunk2008.pdf

[4] Robot/Computer Vision (Florian Klaschka, TU Munich)
http://www9.in.tum.de/seminare/hs.SS01.RV/chapter2.ppt

[5] A Primer on Bezier Curves (Pomax)
https://pomax.github.io/bezierinfo/

[6] Numerical Interpolation: Natural Cubic Spline (Lois Anne Leal)
https://towardsdatascience.com/numerical-interpolation-natural-cubic-spline-52c1157b98ac

[7] Natural Cubic Splines (Arne Morten Kvarving)
https://www.math.ntnu.no/emner/TMA4215/2008h/cubicsplines.pdf

[8] Catmull-Rom splines (Philippe Lucidarme)
https://www.lucidarme.me/catmull-rom-splines/

[9] Cubic Splines (C. Führer)
http://www.maths.lth.se/na/courses/FMN081/FMN081-06/lecture11.pdf

[10] https://commons.wikimedia.org/wiki/File:Hermite_spline_2-segments.svg