Advanced Animation Programming

GPR-450
Daniel S. Buckstein

Speed Control on Curves & Reparameterization
Week 3

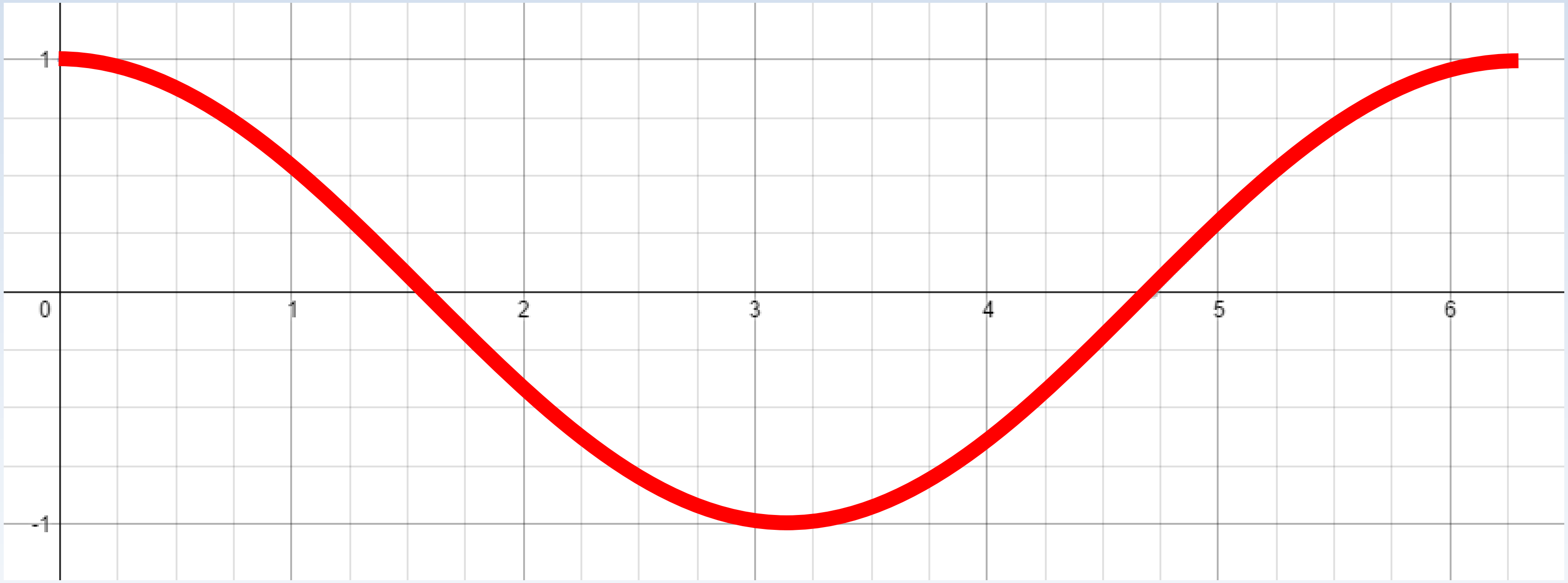
License

Speed Control on Curves

- Arc length of a curve
- Continuous vs. Discrete
  - Discretization of a continuous function
  - Measuring arc length
- The speed control problem
- Speed control algorithm
- Speed profiles

Arc Length of a Curve

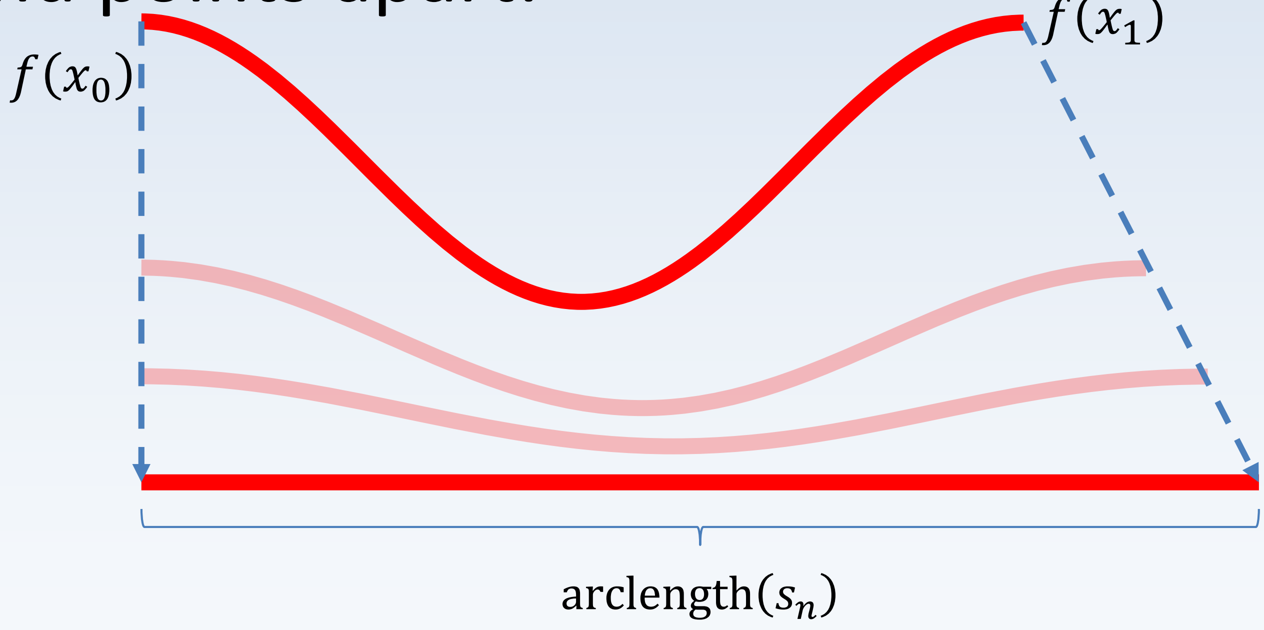
- Continuous function example: $f(x) = \cos(x)$



- What is the *arc length* of the curve?

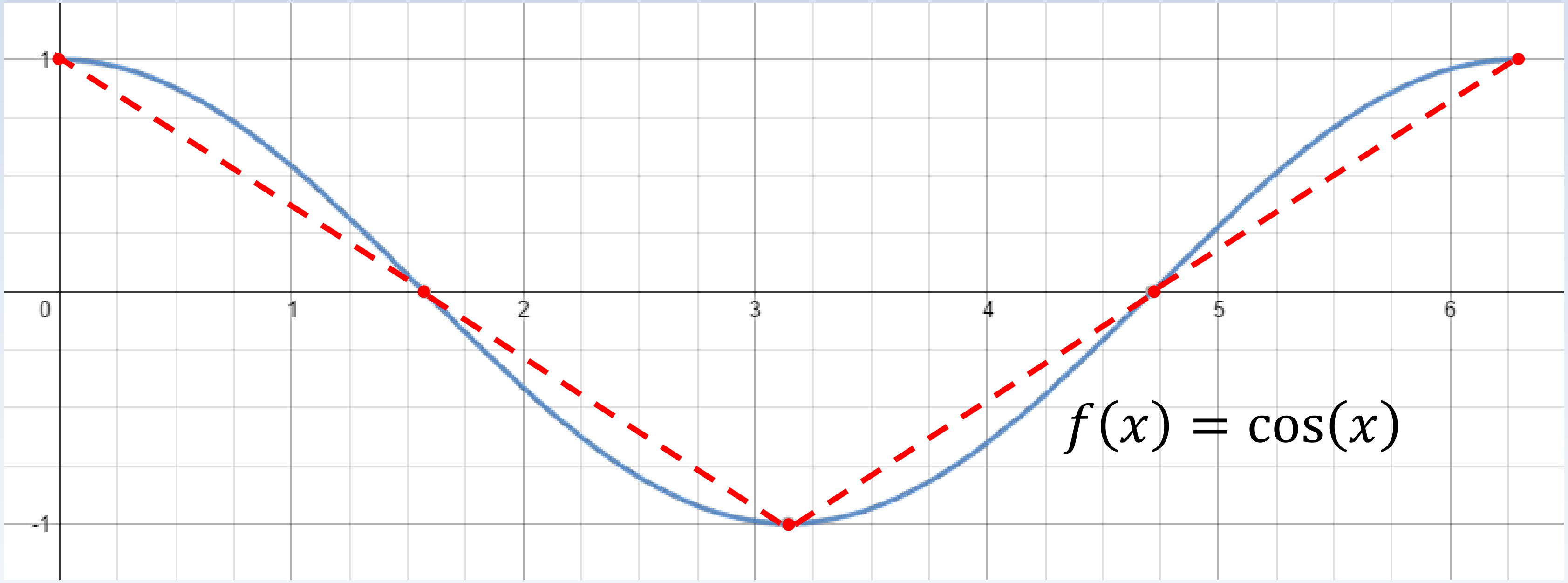
Arc Length of a Curve

- ***Arc length***: the actual *length* of the curve
- Think of it as if we took the curve and pulled the end points apart:



Arc Length of a Curve

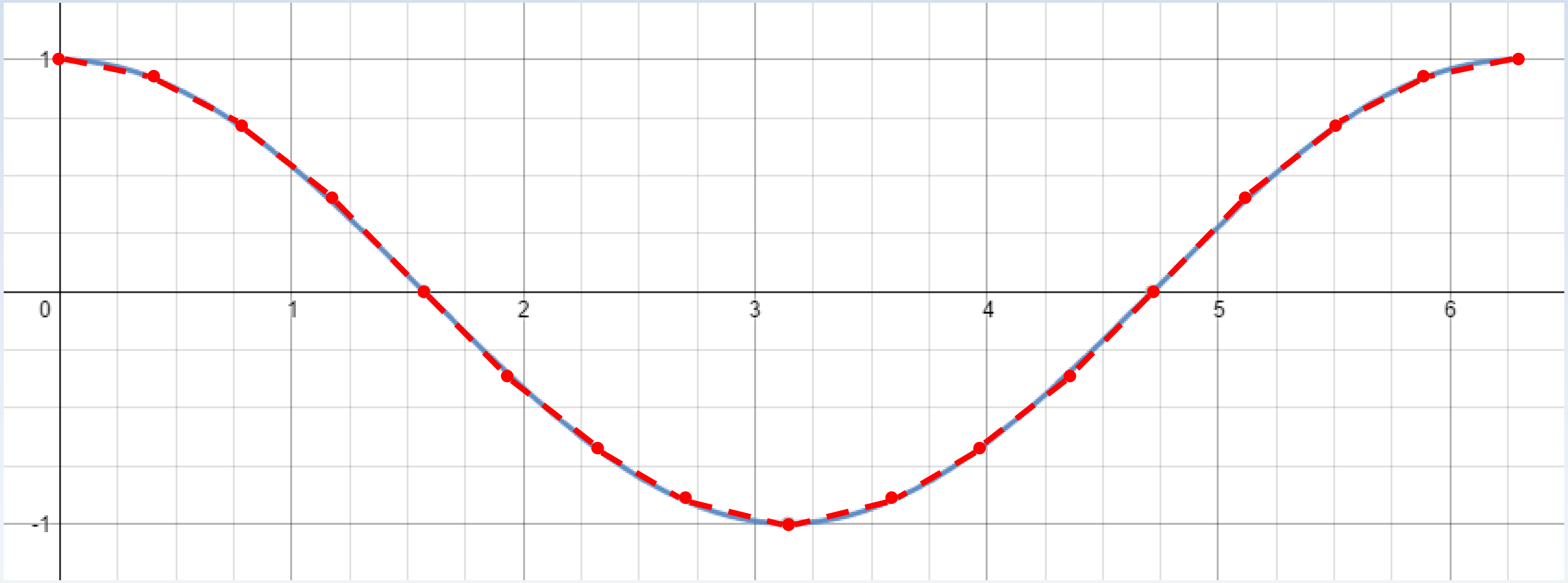
- We don't know all of the values on the curve



- …so we can *sample* to approximate!

Arc Length of a Curve

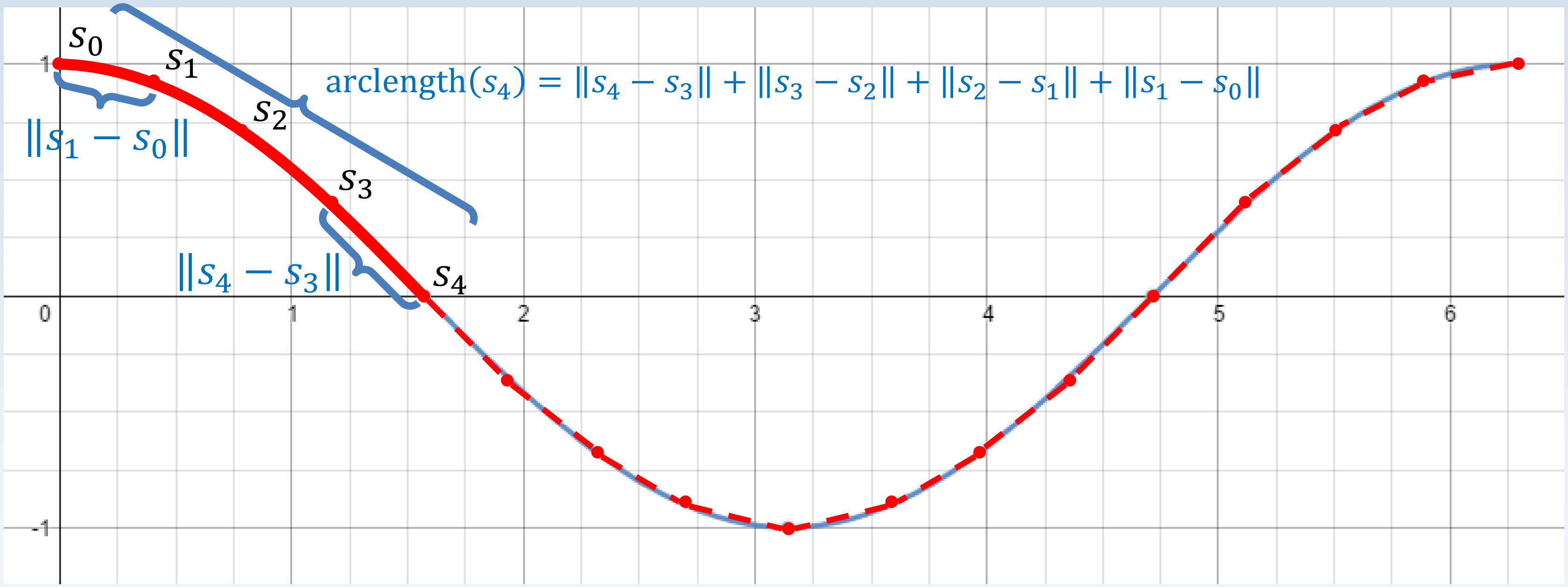
- More samples means a better approximation!



- Arc length = sum of lengths between samples!

Arc Length of a Curve

- More samples means a better approximation!



- The samples are labeled s_i

Arc Length of a Curve

- Algorithm for computing total arc length at any sample on the curve s_i :

$$\mathrm{arclength}(s_i) = \|s_i - s_{i-1}\| + \mathrm{arclength}(s_{i-1})$$
$$\mathrm{arclength}(s_0) = 0$$

- Not a recursive function… just an accumulation of all the arc lengths so far!

Arc Length of a Curve

- Algorithm:

```
float distance = 0; // distance between two samples
sample[0].arclength = 0; // stored per-sample
for (int i = 1; i < numSamples; ++i)
{
    distance = magnitude(sample[i].value
                       – sample[i-1].value);
    sample[i].arclength = distance
                       + sample[i-1].arclength;
}
```

Arc Length of a Curve

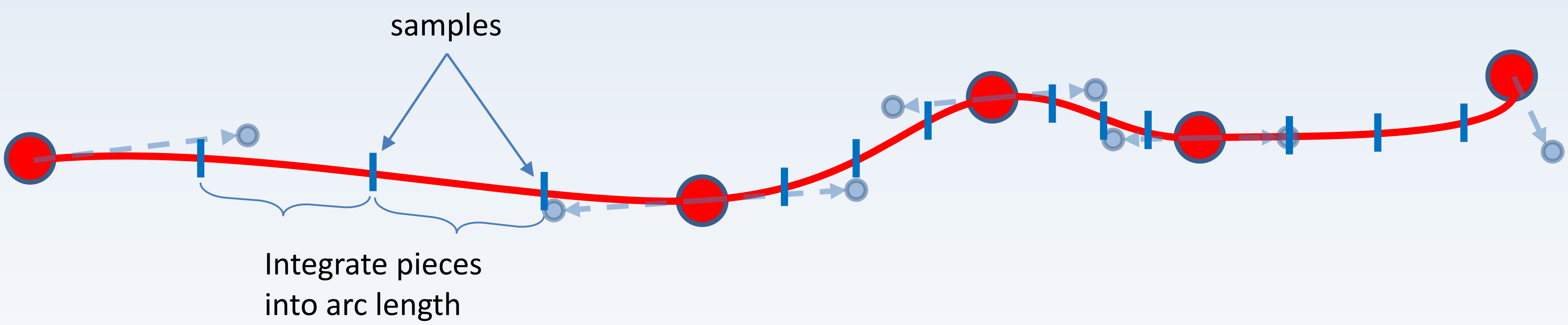
- This process is called *numerical integration*
    - https://en.wikipedia.org/wiki/Numerical_integration
    - https://en.wikipedia.org/wiki/Arc_length

- Continuous integrals don't work in computing

- …so we *discretize* the operation by using *known samples on the curve*

Arc Length of a Curve

- The process is the same for the curves we use in animation:
- Catmull-Rom splines, csplines, Bézier splines…

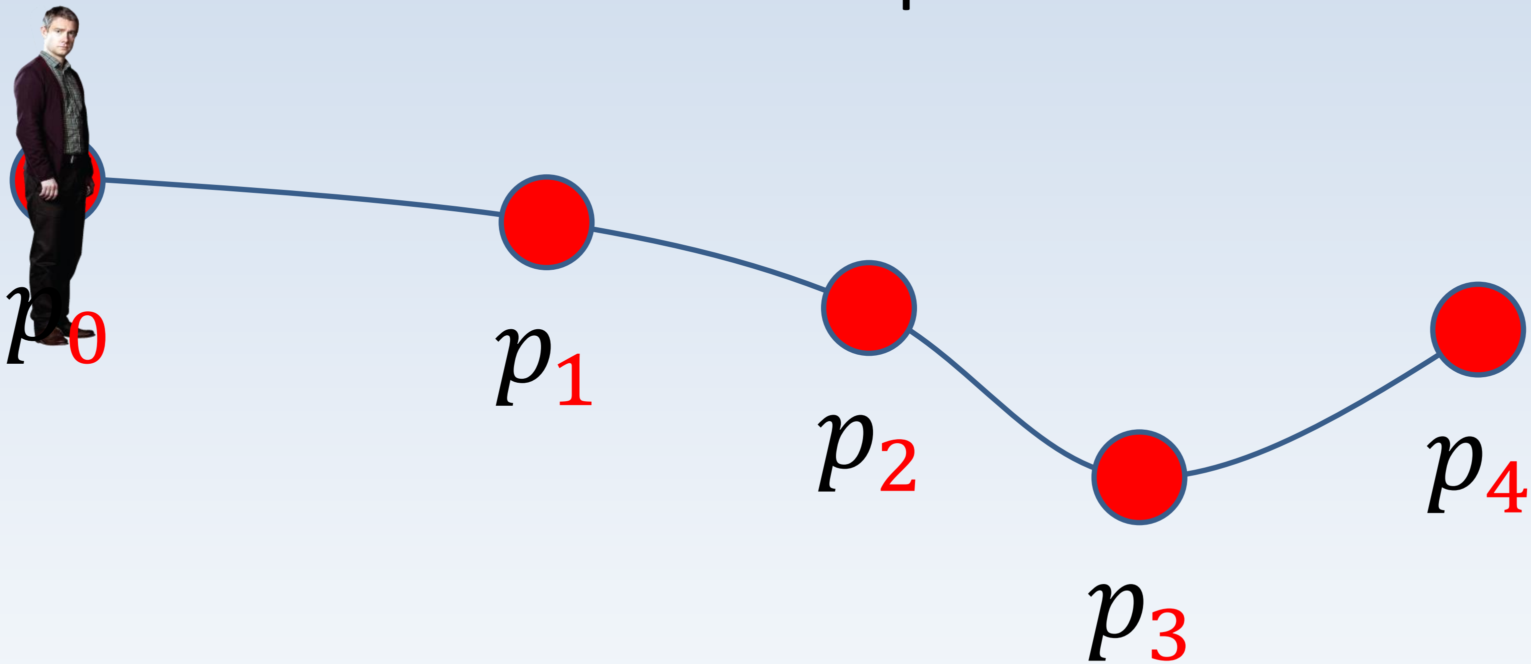


Arc Length of a Curve

- On the topic of *speed control*, why is *arc length* significant???
- Arc length = *distance* travelled along a curve
- Speed = distance / time
- We will ultimately use arc length control the speed of travel along the curve!
- …but what's the actual problem we are solving? (let's find out)

The Speed Control Problem

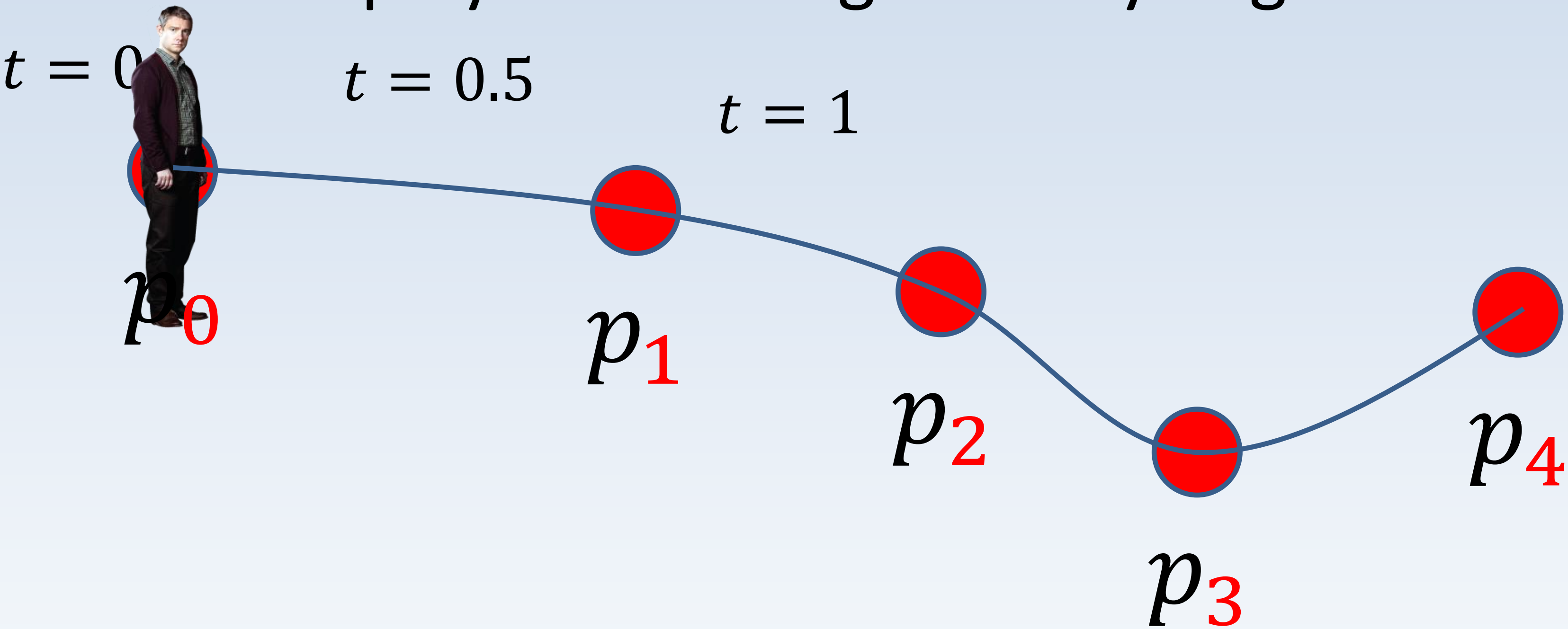
- We want Watson to interpolate on this curve



- …at a constant speed (which he does here)

The Speed Control Problem

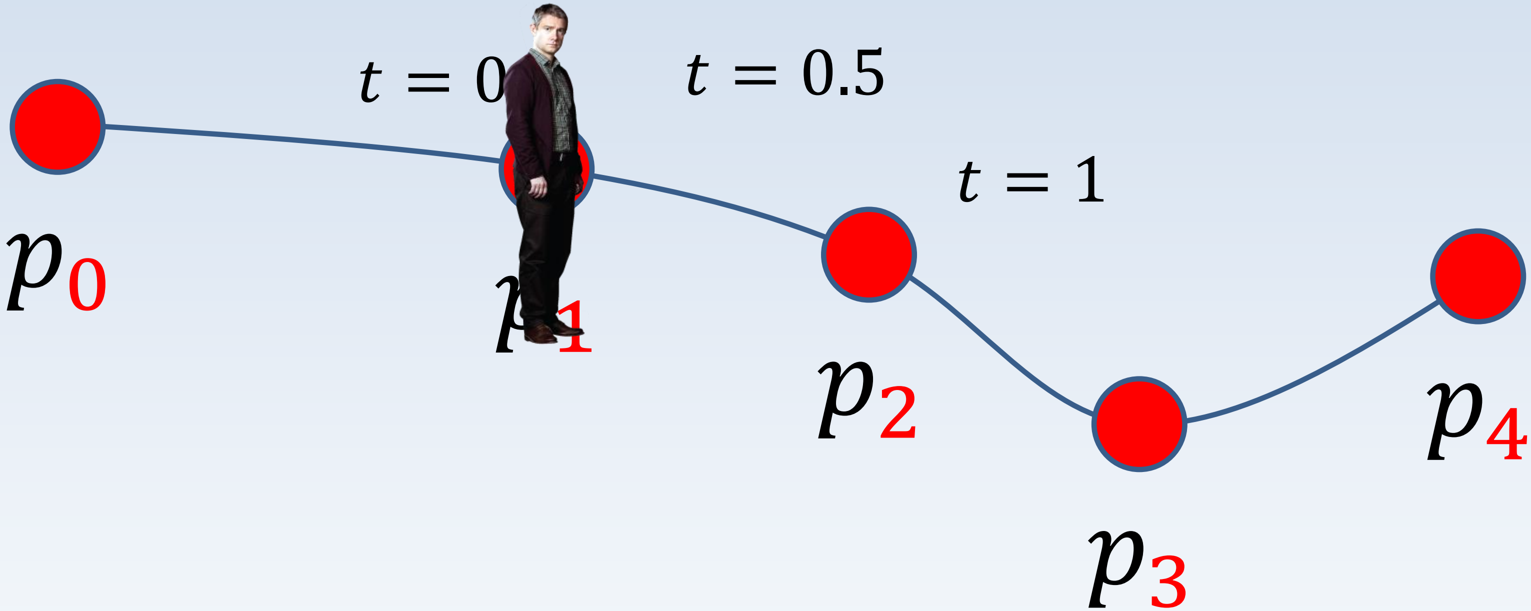
- When played back segment-by-segment… ☹



$p_{\text{Watson}} = \text{CatmullRom}(p_c, p_c, p_{c+1}, p_{c+2}, t)$
$c = 0$
Looping disabled

The Speed Control Problem

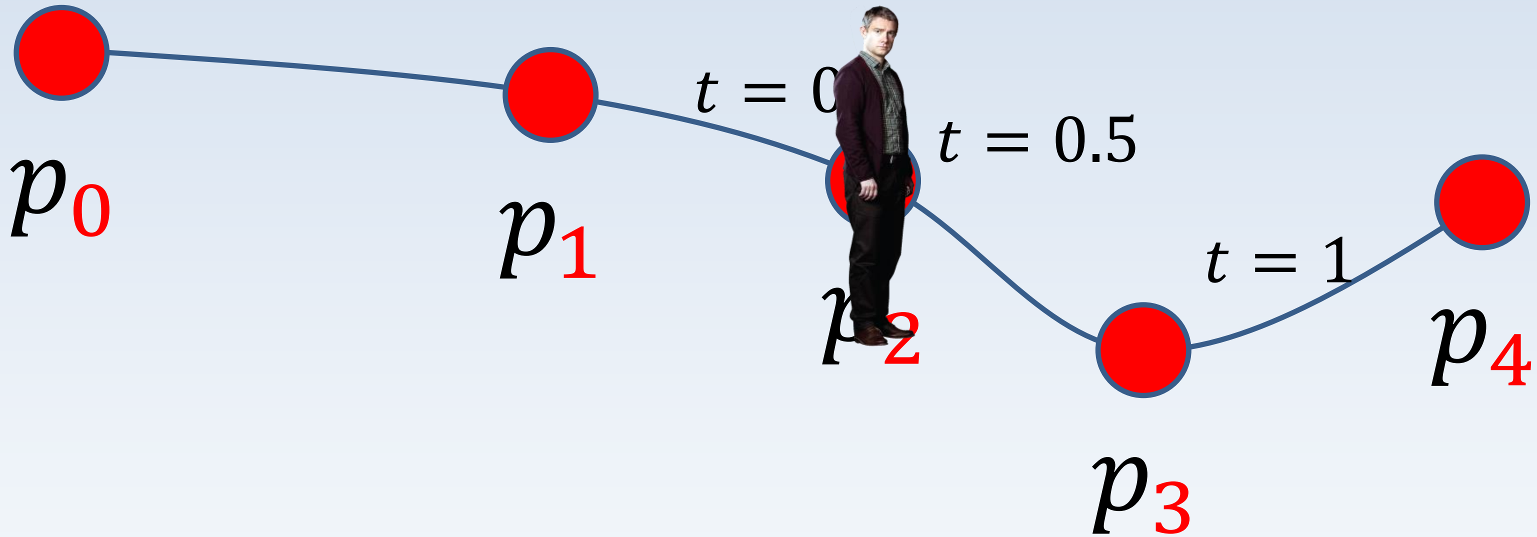
- When played back segment-by-segment… ☹



$p_{\text{Watson}} = \text{CatmullRom}(p_{c-1}, p_c, p_{c+1}, p_{c+2}, t)$
$c = 1$
Looping disabled

The Speed Control Problem

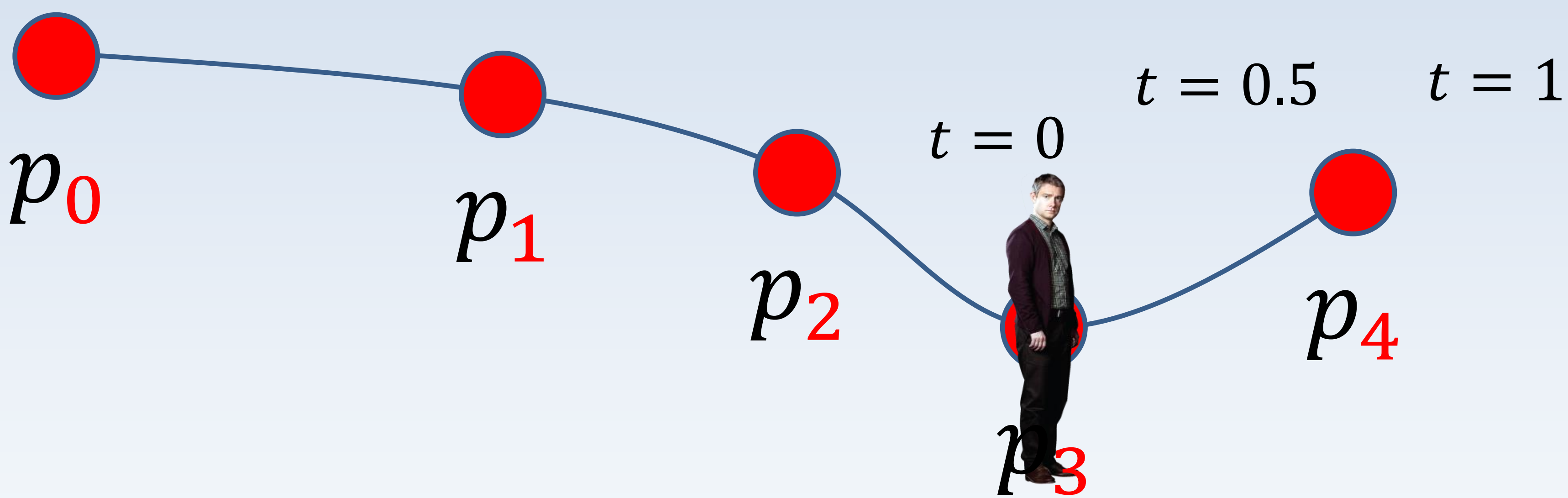
- When played back segment-by-segment… ☹



$p_{\text{Watson}} = \text{CatmullRom}(p_{c-1}, p_c, p_{c+1}, p_{c+2}, t)$
$c = 2$
Looping disabled

The Speed Control Problem

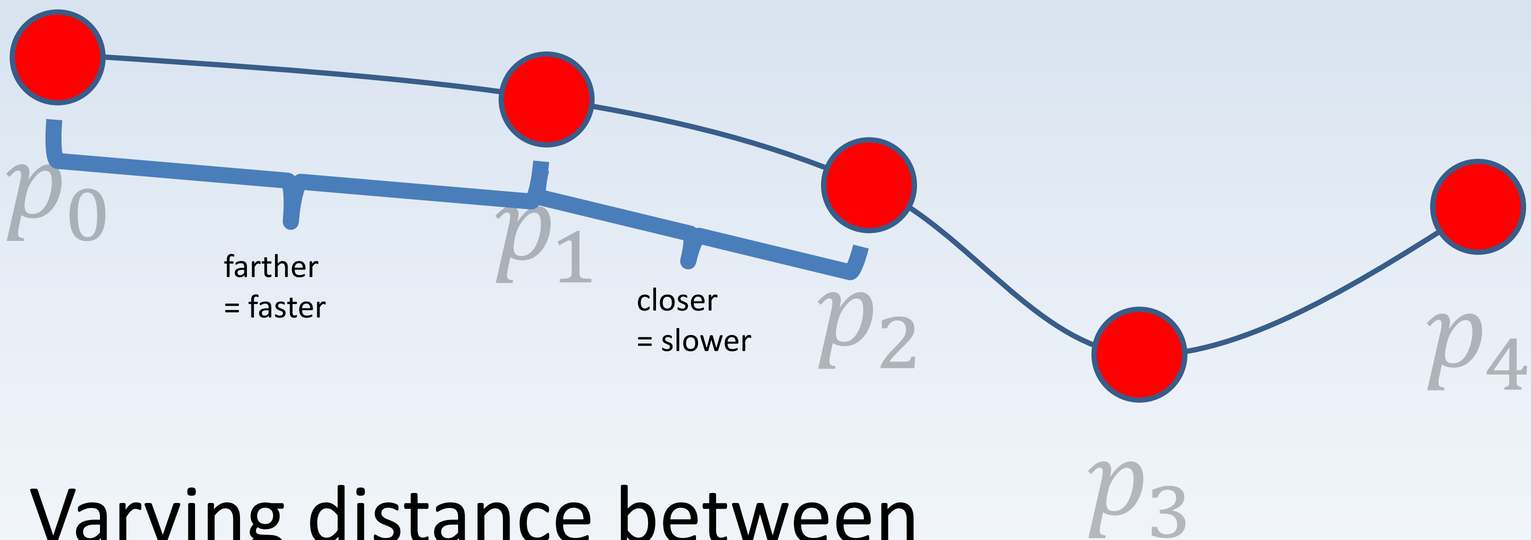
- When played back segment-by-segment… ☹



$p_{\text{Watson}} = \text{CatmullRom}(p_{c-1}, p_c, p_{c+1}, p_{c+1}, t)$
$c = 3$
Looping disabled

The Speed Control Problem

- What's wrong with the way we currently have our system set up? Why???



- Varying distance between samples means varying speed

The Speed Control Problem

- The need for speed! …control!
- ***The speed control problem***:
- If we need to traverse the entire curve in a constant amount of time…
- Longer segments have more distance to cover
- Shorter segments have less distance to cover
- The speed will vary along the curve!

The Speed Control Problem

- If the distance between key points varies…
- How do we adjust speed???

- Measure the *distance*!
- Control interpolation as a function of *distance* instead of *time*!

The Speed Control Problem

- Speed control is often done by *reparametrizing* the entire curve
- We will use the *arc length* (distance) to determine our position along the curve
- Control parameter '*t*' is still used, but our arc length will tell us what to do with it!
- Take many samples per curve segment
- More samples means higher accuracy!

The Speed Control Problem

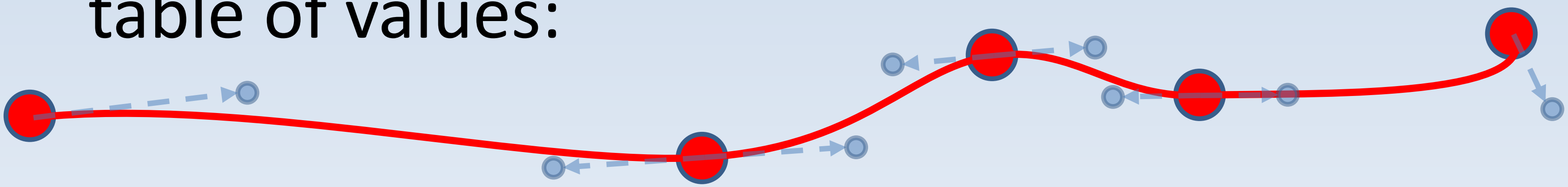
- How do we reparametrize???
- ***Table of values***
- For each sample, store the following info:
- *Segment index* (which *keyframe*)
- *‘t’ value used to acquire sample*
- *Accumulated distance (arc length) at sample*
- *The actual value of the sample*

The Speed Control Problem

- Sampling algorithm:
- For each segment:
  - For the number of samples required per segment:
    - Use interpolation algorithm to compute each sample
    - Compute the accumulated arc length at each sample
      - (see earlier slides)
    - Add entry to table (segment index, interpolation param., arc length, value)

The Speed Control Problem

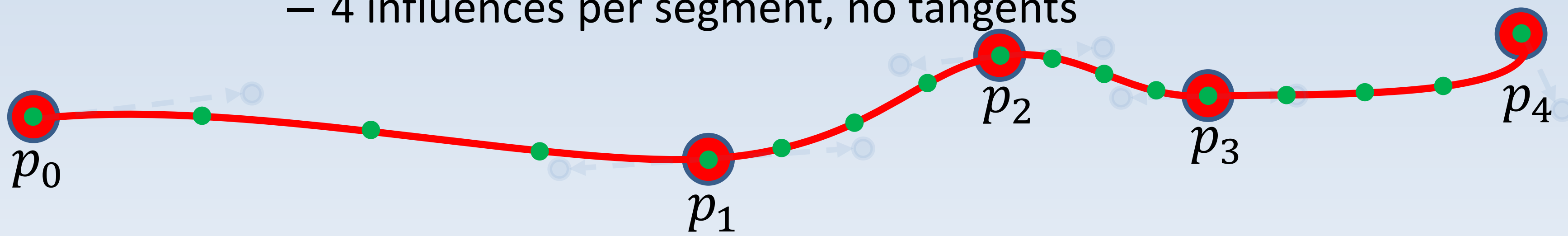
- Example: Let's sample this curve and create a table of values:



- Sampling is done using the interpolation algorithm of your choice… the process is exactly the same!
- We'll figure out how to use the curve later

The Speed Control Problem

- Example: sample curve using Catmull-Rom
  - 4 influences per segment, no tangents

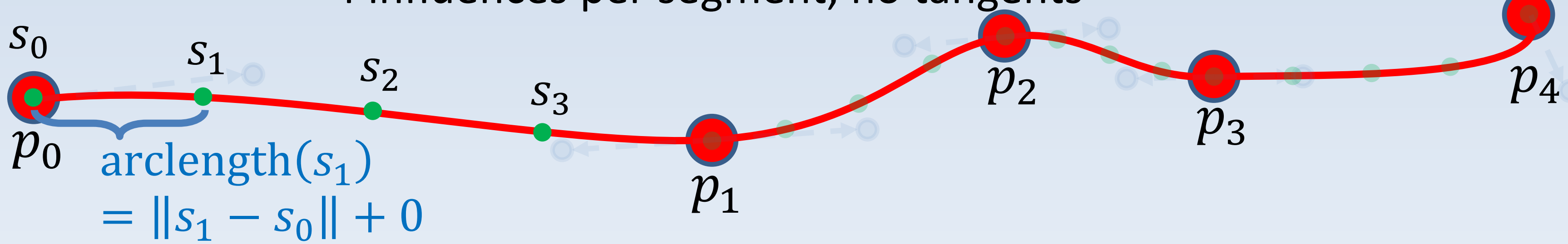


$n = 4$ (samples per segment)

$dt = \frac{1}{n} = \frac{1}{4} = 0.25$ (control value change per sample)

The Speed Control Problem

- Example: sample curve using Catmull-Rom
  - 4 influences per segment, no tangents



Seg. (c)	*t* value	Arc length	Sample
0	0.0	0.0	$s_0 = p_0 =$ (1.0, 1.0)
0	0.25		
0	0.5		
0	0.75		
1	0.0		
1	0.25		
1	0.5		
1	0.75		

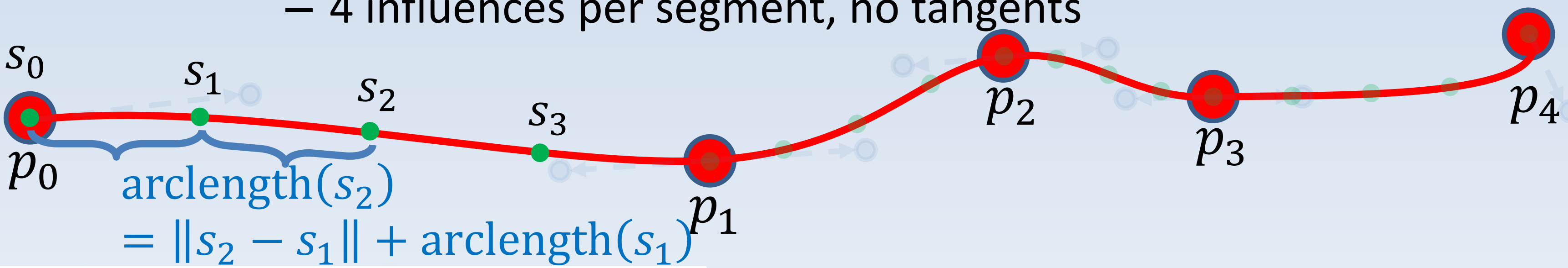
$c = 0$ (segment index… start at first)

$s_1 = \text{CatmullRom}_{p_0 p_0 p_1 p_2}(0.25)$

Example estimates will be used for arc lengths and sample values.

The Speed Control Problem

- Example: sample curve using Catmull-Rom
  - 4 influences per segment, no tangents



Seg. (c)	*t* value	Arc length	Sample
0	0.0	0.0	$s_0 = p_0 =$ (1.0, 1.0)
0	0.25	0.5	$s_1 =$ (1.5, 1.0)
0	0.5		
0	0.75		
1	0.0		
1	0.25		
1	0.5		
1	0.75		

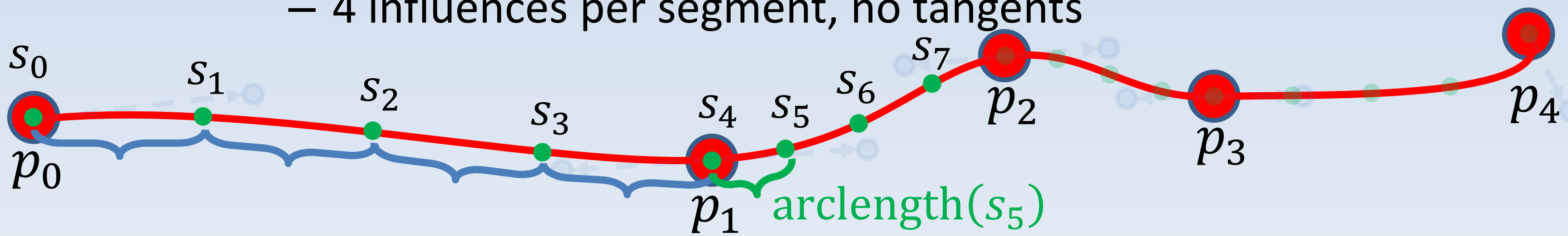
$c = \ 0$ (segment index… start at first)

$s_2 = \mathrm{CatmullRom}_{p_0 p_0 p_1 p_2}(0.50)$

Example estimates will be used for arc lengths and sample values.

The Speed Control Problem

- Example: sample curve using Catmull-Rom
  - 4 influences per segment, no tangents



Seg. (c)	*t* value	Arc length	Sample
0	0.0	0.0	$s_0 = p_0 =$ (1.0, 1.0)
0	0.25	0.5	$s_1 =$ (1.5, 1.0)
0	0.5	1.0025	$s_2 =$ (2.0, 0.95)
0	0.75	1.505	$s_3 =$ (2.5, 0.9)
1	0.0	2.0075	$s_4 = p_1 =$ (3.0, 0.85)
1	0.25	2.2625	$s_5 =$ (3.25, 0.9)
1	0.5	2.5318	$s_6 =$ (3.5, 1.0)
1	0.75	…	$s_7 =$ …

$c = 1$ (segment index)

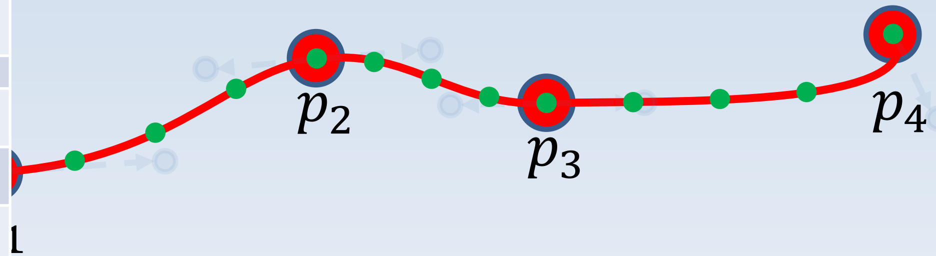
$s_5 = \text{CatmullRom}_{p_0 p_1 p_2 p_3}(0.25)$

This process is repeated for each of samples in each segment.

The Speed Control Problem

rve using Catmull-Rom

Seg. (c)	*t* value	Arc length	Sample
0	0.0	0.0	$s_0 = p_0 = (1.0, 1.0)$
0	0.25	0.5	$s_1 = (1.5, 1.0)$
0	0.5	1.0025	$s_2 = (2.0, 0.95)$
0	0.75	1.505	$s_3 = (2.5, 0.9)$
1	0.0	2.0075	$s_4 = p_1 = (3.0, 0.85)$
1	0.25	2.2625	$s_5 = (3.25, 0.9)$
1	0.5	2.5318	$s_6 = (3.5, 1.0)$
1	0.75	2.8233	$s_7 = (3.75, 1.15)$
2	0.0	3.0926	$s_8 = (4.0, 1.25)$
2	0.25	3.2426	$s_9 = (4.15, 1.25)$
2	0.5	3.3947	$s_{10} = (4.3, 1.225)$
2	0.75	3.5467	$s_{11} = (4.45, 1.2)$
3	0.0	3.6988	$s_{12} = (4.6, 1.175)$
3	0.25	3.9488	$s_{13} = (4.85, 1.175)$
3	0.5	4.1991	$s_{14} = (5.1, 1.1875)$
3	0.75	4.4494	$s_{15} = (5.35, 1.2)$
3	1.0	4.7187	$s_{16} = (5.6, 1.3)$



The table is complete when we have sampled the entire curve!

The Speed Control Problem

- Great news at this point ☺
- Since we know the *total arc length* of the *entire curve*… what might we want to do???
- ***Normalized arc length***: divide all of the arc lengths in the table by total arc length of path
- Now, arc length of 0 represents the beginning of the path, arc length of 1 represents the end

The Speed Control Problem

- ***Normalized arc length***: allows us to have *one* control value for the *entire curve*!!!
- A new '*t*' value that applies to the curve as whole, not just one segment!

Seg. (c)	*t* value	Arc length	Sample
0	0.0	0.0	$s_0 = p_0 =$ (1.0, 1.0)
0	0.25	0.1059	$s_1 =$ (1.5, 1.0)
0	0.5	0.2124	$s_2 =$ (2.0, 0.95)
0	0.75	0.3189	$s_3 =$ (2.5, 0.9)
1	0.0	0.4254	$s_4 = p_1 =$ (3.0, 0.85)
1	0.25	0.4795	$s_5 =$ (3.25, 0.9)
1	0.5	0.5365	$s_6 =$ (3.5, 1.0)
1	0.75	0.5983	$s_7 =$ (3.75, 1.15)
2	0.0	0.6554	$s_8 =$ (4.0, 1.25)
2	0.25	0.6872	$s_9 =$ (4.15, 1.25)
2	0.5	0.7194	$s_{10} =$ (4.3, 1.225)
2	0.75	0.7516	$s_{11} =$ (4.45, 1.2)
3	0.0	0.7839	$s_{12} =$ (4.6, 1.175)
3	0.25	0.8368	$s_{13} =$ (4.85, 1.175)
3	0.5	0.8899	$s_{14} =$ (5.1, 1.1875)
3	0.75	0.9429	$s_{15} =$ (5.35, 1.2)
3	1.0	1.0	$s_{16} =$ (5.6, 1.3)

The Speed Control Problem

- Hooray!!! We sampled the curve!!!

- But wait… how do we interpolate it now using the samples we've got and arc length?

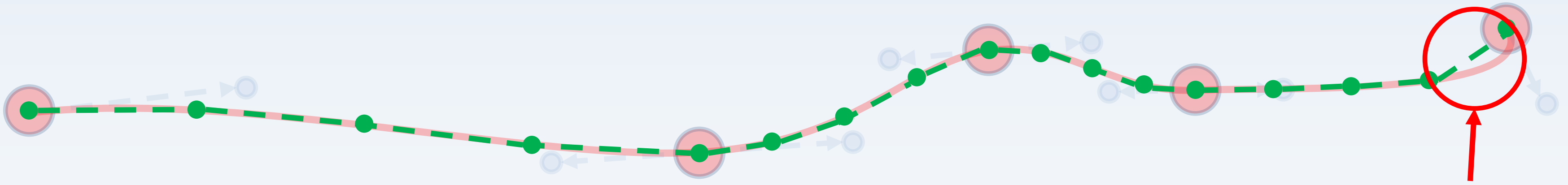
Speed Control on Curves

- Now we have the entire curve reparametrized as a function of *arc length* or *distance*
- Directly related to speed…
- …but we need to be able to interpolate along the curve using this new control value!

- *Two methods of interpolation!*

Speed Control on Curves

- ***Method 1***: using only the data we have, the *type of curve* is no longer relevant…
- …we have a large set of discrete samples
- …the entire curve can be *approximated* as a set of ***line segments***:



(the operative word is “*approximate*”)

Speed Control on Curves

- ***Method 1***: Given only a *normalized arc length* as input, we can look up that value in our table to determine which sample is closest!
- ***Linear interpolate*** between the samples that enclose our desired arc length value!!!

- Example: we want the point where arc length is ***0.7***

Speed Control on Curves

- ***Method 1: LERP***
- Example: desired arc length = 0.7
- Look it up in the table:

Seg. (c)	*t* value	Arc length	Sample
0	0.0	0.0	$s_0 = p_0 = (1.0, 1.0)$
0	0.25	0.1059	$s_1 = (1.5, 1.0)$
0	0.5	0.2124	$s_2 = (2.0, 0.95)$
0	0.75	0.3189	$s_3 = (2.5, 0.9)$
1	0.0	0.4254	$s_4 = p_1 = (3.0, 0.85)$
1	0.25	0.4795	$s_5 = (3.25, 0.9)$
1	0.5	0.5365	$s_6 = (3.5, 1.0)$
1	0.75	0.5983	$s_7 = (3.75, 1.15)$
2	0.0	0.6554	$s_8 = (4.0, 1.25)$
2	0.25	0.6872	$s_9 = (4.15, 1.25)$
2	0.5	0.7194	$s_{10} = (4.3, 1.225)$
2	0.75	0.7516	$s_{11} = (4.45, 1.2)$
3	0.0	0.7839	$s_{12} = (4.6, 1.175)$
3	0.25	0.8368	$s_{13} = (4.85, 1.175)$
3	0.5	0.8899	$s_{14} = (5.1, 1.1875)$
3	0.75	0.9429	$s_{15} = (5.35, 1.2)$
3	1.0	1.0	$s_{16} = (5.6, 1.3)$

- 0.7 lies somewhere between samples 9 and 10

Speed Control on Curves

- A new problem is introduced:
- Where is the *arc length* of 0.7 ***relative*** to the surrounding sampled arc lengths?
- How do we solve this???
- LERP is used to find a point between two key values given a normalized, relative '*t*' value…
- … but what we want is the opposite: we want the *t* value given a known result…

Speed Control on Curves

- The function to tell us where our requested value is *relative to* the two other values…
- …is actually the inverse of LERP!

$$\begin{aligned} v &= \mathrm{lerp}_{v_0 v_1}(t) = (1 - t)v_0 + t\, v_1 \\ &= v_0 + t(v_1 - v_0) \end{aligned}$$

- By isolating *t* we find the inverse function:

$$\boldsymbol{t = \mathbf{lerp}^{-1}_{v_0 v_1}(v) = \frac{v - v_0}{v_1 - v_0}}$$

Speed Control on Curves

- ***Method 1: LERP***
- Where is 0.7 relative to the *known arc lengths*?

$$t = \mathrm{lerp}_{v_0 v_1}^{-1}(v)$$

$v = 0.7$ (desired value)

$v_0 = 0.6872$ (first input value)

$v_1 = 0.7194$ (second input)

Seg. (c)	*t* value	Arc length	Sample
0	0.0	0.0	$s_0 = p_0 = (1.0, 1.0)$
0	0.25	0.1059	$s_1 = (1.5, 1.0)$
0	0.5	0.2124	$s_2 = (2.0, 0.95)$
0	0.75	0.3189	$s_3 = (2.5, 0.9)$
1	0.0	0.4254	$s_4 = p_1 = (3.0, 0.85)$
1	0.25	0.4795	$s_5 = (3.25, 0.9)$
1	0.5	0.5365	$s_6 = (3.5, 1.0)$
1	0.75	0.5983	$s_7 = (3.75, 1.15)$
2	0.0	0.6554	$s_8 = (4.0, 1.25)$
2	0.25	0.6872	$s_9 = (4.15, 1.25)$
2	0.5	0.7194	$s_{10} = (4.3, 1.225)$
2	0.75	0.7516	$s_{11} = (4.45, 1.2)$
3	0.0	0.7839	$s_{12} = (4.6, 1.175)$
3	0.25	0.8368	$s_{13} = (4.85, 1.175)$
3	0.5	0.8899	$s_{14} = (5.1, 1.1875)$
3	0.75	0.9429	$s_{15} = (5.35, 1.2)$
3	1.0	1.0	$s_{16} = (5.6, 1.3)$

Speed Control on Curves

- ***Method 1: LERP***
- Where is 0.7 relative to the *known arc lengths*?

$$t = \mathrm{lerp}_{v_0 v_1}^{-1}(v)$$
$$= \frac{0.7 - 0.6872}{0.7194 - 0.6872}$$
$$= \frac{0.0128}{0.0322}$$
$$t \approx \mathbf{0.3975}$$

Seg. (c)	*t* value	Arc length	Sample
0	0.0	0.0	$s_0 = p_0 = (1.0, 1.0)$
0	0.25	0.1059	$s_1 = (1.5, 1.0)$
0	0.5	0.2124	$s_2 = (2.0, 0.95)$
0	0.75	0.3189	$s_3 = (2.5, 0.9)$
1	0.0	0.4254	$s_4 = p_1 = (3.0, 0.85)$
1	0.25	0.4795	$s_5 = (3.25, 0.9)$
1	0.5	0.5365	$s_6 = (3.5, 1.0)$
1	0.75	0.5983	$s_7 = (3.75, 1.15)$
2	0.0	0.6554	$s_8 = (4.0, 1.25)$
2	0.25	$\boxed{0.6872}$	$\boxed{s_9 = (4.15, 1.25)}$
2	0.5	$\boxed{0.7194}$	$\boxed{s_{10} = (4.3, 1.225)}$
2	0.75	0.7516	$s_{11} = (4.45, 1.2)$
3	0.0	0.7839	$s_{12} = (4.6, 1.175)$
3	0.25	0.8368	$s_{13} = (4.85, 1.175)$
3	0.5	0.8899	$s_{14} = (5.1, 1.1875)$
3	0.75	0.9429	$s_{15} = (5.35, 1.2)$
3	1.0	1.0	$s_{16} = (5.6, 1.3)$

Speed Control on Curves

- ***Method 1: LERP***
- Use our t value to LERP between the known sample values!!!

$$p = \mathrm{lerp}_{s_9 s_{10}}(0.3975)$$
$$= (1 - 0.3975)s_9 + (0.3975)s_{10}$$
$$= 0.6025 \begin{bmatrix} 4.15 \\ 1.25 \end{bmatrix} + 0.3975 \begin{bmatrix} 4.3 \\ 1.225 \end{bmatrix}$$

$$p = \begin{bmatrix} 2.500375 \\ 0.753125 \end{bmatrix} + \begin{bmatrix} 1.70925 \\ 0.4869375 \end{bmatrix}$$

$$\boldsymbol{p} \approx \begin{bmatrix} \mathbf{4.21} \\ \mathbf{1.24} \end{bmatrix}$$

Seg. (c)	*t* value	Arc length	Sample
0	0.0	0.0	$s_0 = p_0 = (1.0, 1.0)$
0	0.25	0.1059	$s_1 = (1.5, 1.0)$
0	0.5	0.2124	$s_2 = (2.0, 0.95)$
0	0.75	0.3189	$s_3 = (2.5, 0.9)$
1	0.0	0.4254	$s_4 = p_1 = (3.0, 0.85)$
1	0.25	0.4795	$s_5 = (3.25, 0.9)$
1	0.5	0.5365	$s_6 = (3.5, 1.0)$
1	0.75	0.5983	$s_7 = (3.75, 1.15)$
2	0.0	0.6554	$s_8 = (4.0, 1.25)$
2	0.25	0.6872	$s_9 = (4.15, 1.25)$
2	0.5	0.7194	$s_{10} = (4.3, 1.225)$
2	0.75	0.7516	$s_{11} = (4.45, 1.2)$
3	0.0	0.7839	$s_{12} = (4.6, 1.175)$
3	0.25	0.8368	$s_{13} = (4.85, 1.175)$
3	0.5	0.8899	$s_{14} = (5.1, 1.1875)$
3	0.75	0.9429	$s_{15} = (5.35, 1.2)$
3	1.0	1.0	$s_{16} = (5.6, 1.3)$

Speed Control on Curves

- ***Method 1: LERP*** (visualized)
- Locating where the arc length is 0.7

$\text{arclength}(s_9)$
$= 0.6872$

$\text{arclength}(s_{10})$
$= 0.7194$

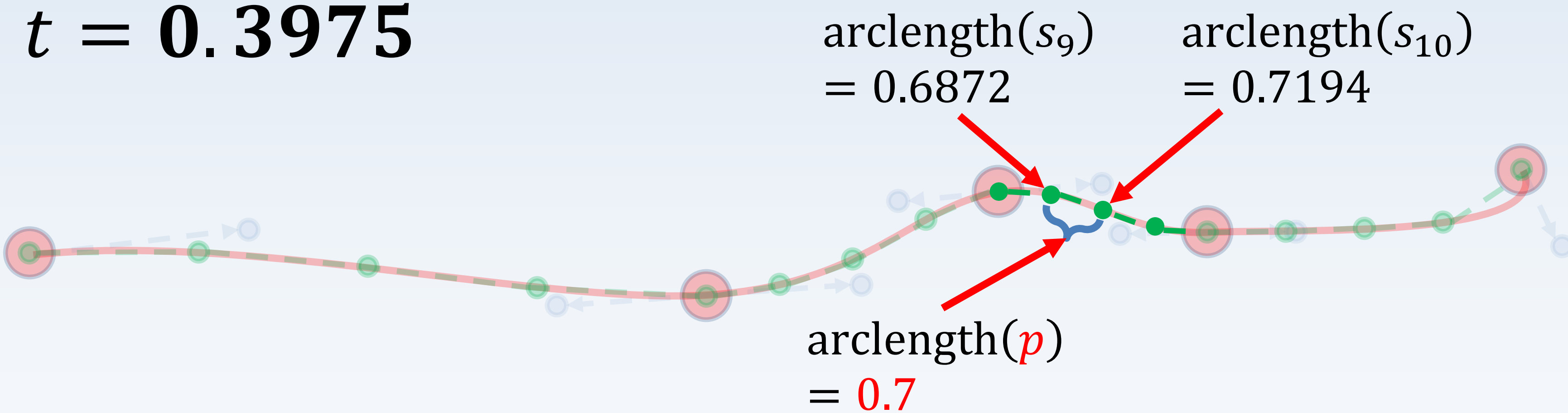
$\text{arclength}(p)$
$= 0.7$ (somewhere between s_9 and s_{10})

Speed Control on Curves

- ***Method 1: LERP*** (visualized)
- Need the *relative* position of 0.7 between the other two values: *inverse LERP*

$t = \mathrm{lerp}^{-1}(0.6872, 0.7194, 0.7)$

$\boldsymbol{t = 0.3975}$



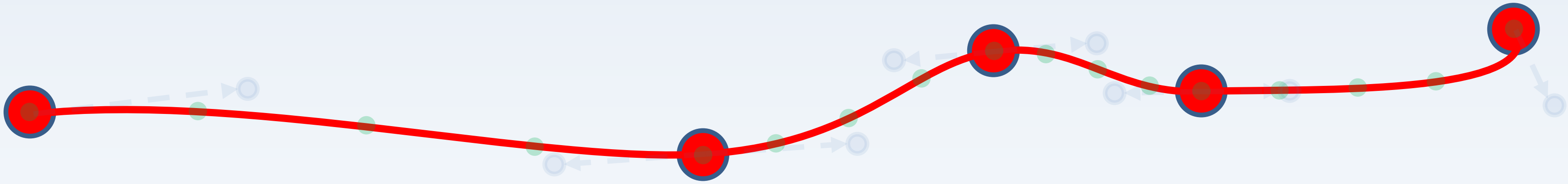
$\mathbf{0.3975}$ of the way between s_9 and s_{10}

Speed Control on Curves

- ***Method 1: LERP*** (visualized)
- Use the *t* value we just found to compute the actual value between the two samples!!!
- $p = \mathrm{lerp}_{s_9 s_{10}}(0.3975) \approx (4.21, 1.24)$

Speed Control on Curves

- ***Method 2***: The *type of curve* is still relevant
- …and we are going to interpolate using the proper algorithm for higher precision!
- This method is similar to the other, but we use different values in our table…???



Speed Control on Curves

- ***Method 2: Spline***
- An additional piece of information is needed:
- We must store the segments' end points!
- Just duplicate the first sample from the next segment!

Seg. (c)	*t* value	Arc length	Sample
0	0.0	0.0	$s_0 = p_0 = (1.0, 1.0)$
0	0.25	0.1059	$s_1 = (1.5, 1.0)$
0	0.5	0.2124	$s_2 = (2.0, 0.95)$
0	0.75	0.3189	$s_3 = (2.5, 0.9)$
0	1.0		
1	0.0	0.4254	$s_4 = p_1 = (3.0, 0.85)$
1	0.25	0.4795	$s_5 = (3.25, 0.9)$
1	0.5	0.5365	$s_6 = (3.5, 1.0)$
1	0.75	0.5983	$s_7 = (3.75, 1.15)$
1	1.0		
2	0.0	0.6554	$s_8 = (4.0, 1.25)$
2	0.25	0.6872	$s_9 = (4.15, 1.25)$
2	0.5	0.7194	$s_{10} = (4.3, 1.225)$
2	0.75	0.7516	$s_{11} = (4.45, 1.2)$
2	1.0		
3	0.0	0.7839	$s_{12} = (4.6, 1.175)$
3	0.25	0.8368	$s_{13} = (4.85, 1.175)$
3	0.5	0.8899	$s_{14} = (5.1, 1.1875)$
3	0.75	0.9429	$s_{15} = (5.35, 1.2)$
3	1.0	1.0	$s_{16} = (5.6, 1.3)$

Speed Control on Curves

- ***Method 2: Spline***
- Example: find the point where arc length is ***0.7***
- The first step is exactly the same: look up the desired arc length in the table!

Seg. (c)	*t* value	Arc length	Sample
0	0.0	0.0	$s_0 = p_0 = (1.0, 1.0)$
0	0.25	0.1059	$s_1 = (1.5, 1.0)$
0	0.5	0.2124	$s_2 = (2.0, 0.95)$
0	0.75	0.3189	$s_3 = (2.5, 0.9)$
0	1.0	0.4254	$s_4 = p_1 = (3.0, 0.85)$
1	0.0	0.4254	$s_4 = p_1 = (3.0, 0.85)$
1	0.25	0.4795	$s_5 = (3.25, 0.9)$
1	0.5	0.5365	$s_6 = (3.5, 1.0)$
1	0.75	0.5983	$s_7 = (3.75, 1.15)$
1	1.0	0.6554	$s_8 = (4.0, 1.25)$
2	0.0	0.6554	$s_8 = (4.0, 1.25)$
2	0.25	0.6872	$s_9 = (4.15, 1.25)$
2	0.5	0.7194	$s_{10} = (4.3, 1.225)$
2	0.75	0.7516	$s_{11} = (4.45, 1.2)$
2	1.0	0.7839	$s_{12} = (4.6, 1.175)$
3	0.0	0.7839	$s_{12} = (4.6, 1.175)$
3	0.25	0.8368	$s_{13} = (4.85, 1.175)$
3	0.5	0.8899	$s_{14} = (5.1, 1.1875)$
3	0.75	0.9429	$s_{15} = (5.35, 1.2)$
3	1.0	1.0	$s_{16} = (5.6, 1.3)$

Speed Control on Curves

- ***Method 2: Spline***
- The next step is also exactly the same:
- Need to know where 0.7 is relative to the known arc lengths:

$$t = \mathrm{lerp}^{-1}_{v_0 v_1}(0.7)$$
$$= 0.3975$$

Seg. (c)	*t* value	Arc length	Sample
0	0.0	0.0	$s_0 = p_0 = (1.0, 1.0)$
0	0.25	0.1059	$s_1 = (1.5, 1.0)$
0	0.5	0.2124	$s_2 = (2.0, 0.95)$
0	0.75	0.3189	$s_3 = (2.5, 0.9)$
0	1.0	0.4254	$s_4 = p_1 = (3.0, 0.85)$
1	0.0	0.4254	$s_4 = p_1 = (3.0, 0.85)$
1	0.25	0.4795	$s_5 = (3.25, 0.9)$
1	0.5	0.5365	$s_6 = (3.5, 1.0)$
1	0.75	0.5983	$s_7 = (3.75, 1.15)$
1	1.0	0.6554	$s_8 = (4.0, 1.25)$
2	0.0	0.6554	$s_8 = (4.0, 1.25)$
2	0.25	0.6872	$s_9 = (4.15, 1.25)$
2	0.5	0.7194	$s_{10} = (4.3, 1.225)$
2	0.75	0.7516	$s_{11} = (4.45, 1.2)$
2	1.0	0.7839	$s_{12} = (4.6, 1.175)$
3	0.0	0.7839	$s_{12} = (4.6, 1.175)$
3	0.25	0.8368	$s_{13} = (4.85, 1.175)$
3	0.5	0.8899	$s_{14} = (5.1, 1.1875)$
3	0.75	0.9429	$s_{15} = (5.35, 1.2)$
3	1.0	1.0	$s_{16} = (5.6, 1.3)$

Speed Control on Curves

- ***Method 2: Spline***
- The next step is a bit different…
- The first method *approximates* with LERP
- Now we want precision using our spline interpolation algorithm
- What do we do next???

Speed Control on Curves

- ***Method 2: Spline***
- Next: determine the ***segment*** that our desired arc length lies within
- Instead of using our *t* value to interpolate the raw samples…

Seg. (c)	*t* value	Arc length	Sample
0	0.0	0.0	$s_0 = p_0 = (1.0, 1.0)$
0	0.25	0.1059	$s_1 = (1.5, 1.0)$
0	0.5	0.2124	$s_2 = (2.0, 0.95)$
0	0.75	0.3189	$s_3 = (2.5, 0.9)$
0	1.0	0.4254	$s_4 = p_1 = (3.0, 0.85)$
1	0.0	0.4254	$s_4 = p_1 = (3.0, 0.85)$
1	0.25	0.4795	$s_5 = (3.25, 0.9)$
1	0.5	0.5365	$s_6 = (3.5, 1.0)$
1	0.75	0.5983	$s_7 = (3.75, 1.15)$
1	1.0	0.6554	$s_8 = (4.0, 1.25)$
2	0.0	0.6554	$s_8 = (4.0, 1.25)$
2	0.25	0.6872	$s_9 = (4.15, 1.25)$
2	0.5	0.7194	$s_{10} = (4.3, 1.225)$
2	0.75	0.7516	$s_{11} = (4.45, 1.2)$
2	1.0	0.7839	$s_{12} = (4.6, 1.175)$
3	0.0	0.7839	$s_{12} = (4.6, 1.175)$
3	0.25	0.8368	$s_{13} = (4.85, 1.175)$
3	0.5	0.8899	$s_{14} = (5.1, 1.1875)$
3	0.75	0.9429	$s_{15} = (5.35, 1.2)$
3	1.0	1.0	$s_{16} = (5.6, 1.3)$

Speed Control on Curves

- ***Method 2: Spline***
- …we compute a *spline interpolation parameter!!!*
- We will use this value to perform spline interpolation on the correct segment!!!

Seg. (c)	*t* value	Arc length	Sample
0	0.0	0.0	$s_0 = p_0 = (1.0, 1.0)$
0	0.25	0.1059	$s_1 = (1.5, 1.0)$
0	0.5	0.2124	$s_2 = (2.0, 0.95)$
0	0.75	0.3189	$s_3 = (2.5, 0.9)$
0	1.0	0.4254	$s_4 = p_1 = (3.0, 0.85)$
1	0.0	0.4254	$s_4 = p_1 = (3.0, 0.85)$
1	0.25	0.4795	$s_5 = (3.25, 0.9)$
1	0.5	0.5365	$s_6 = (3.5, 1.0)$
1	0.75	0.5983	$s_7 = (3.75, 1.15)$
1	1.0	0.6554	$s_8 = (4.0, 1.25)$
2	0.0	0.6554	$s_8 = (4.0, 1.25)$
2	**0.25**	0.6872	$s_9 = (4.15, 1.25)$
2	**0.5**	0.7194	$s_{10} = (4.3, 1.225)$
2	0.75	0.7516	$s_{11} = (4.45, 1.2)$
2	1.0	0.7839	$s_{12} = (4.6, 1.175)$
3	0.0	0.7839	$s_{12} = (4.6, 1.175)$
3	0.25	0.8368	$s_{13} = (4.85, 1.175)$
3	0.5	0.8899	$s_{14} = (5.1, 1.1875)$
3	0.75	0.9429	$s_{15} = (5.35, 1.2)$
3	1.0	1.0	$s_{16} = (5.6, 1.3)$

Speed Control on Curves

- ***Method 2: Spline***
- We'll call our new parameter t'
- Interpolate between the t values that we used to acquire the respective samples!!!

Seg. (c)	*t* value	Arc length	Sample
0	0.0	0.0	$s_0 = p_0 = (1.0, 1.0)$
0	0.25	0.1059	$s_1 = (1.5, 1.0)$
0	0.5	0.2124	$s_2 = (2.0, 0.95)$
0	0.75	0.3189	$s_3 = (2.5, 0.9)$
0	1.0	0.4254	$s_4 = p_1 = (3.0, 0.85)$
1	0.0	0.4254	$s_4 = p_1 = (3.0, 0.85)$
1	0.25	0.4795	$s_5 = (3.25, 0.9)$
1	0.5	0.5365	$s_6 = (3.5, 1.0)$
1	0.75	0.5983	$s_7 = (3.75, 1.15)$
1	1.0	0.6554	$s_8 = (4.0, 1.25)$
2	0.0	0.6554	$s_8 = (4.0, 1.25)$
2	**0.25**	0.6872	$s_9 = (4.15, 1.25)$
2	**0.5**	0.7194	$s_{10} = (4.3, 1.225)$
2	0.75	0.7516	$s_{11} = (4.45, 1.2)$
2	1.0	0.7839	$s_{12} = (4.6, 1.175)$
3	0.0	0.7839	$s_{12} = (4.6, 1.175)$
3	0.25	0.8368	$s_{13} = (4.85, 1.175)$
3	0.5	0.8899	$s_{14} = (5.1, 1.1875)$
3	0.75	0.9429	$s_{15} = (5.35, 1.2)$
3	1.0	1.0	$s_{16} = (5.6, 1.3)$

Speed Control on Curves

- ***Method 2: Spline***
- Compute t' :

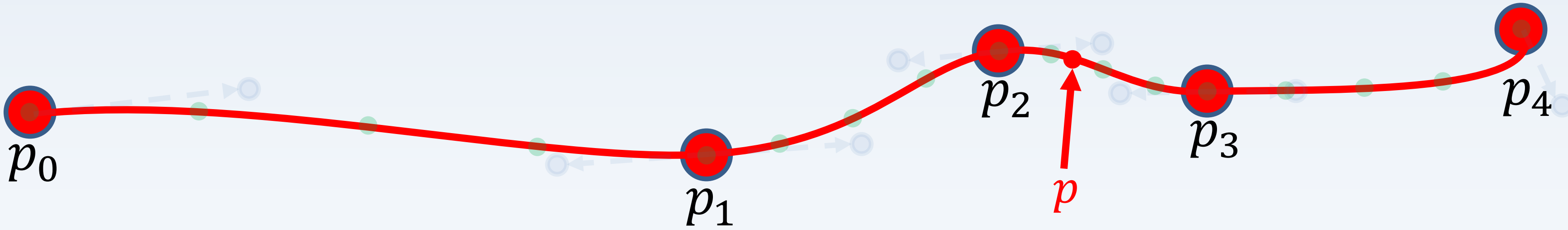
$$t' = \mathrm{lerp}_{t_9 t_{10}}(0.3975)$$
$$= (1 - 0.3975)0.25 + (0.3975)0.5$$
$$= 0.150625 + 0.19875$$
$$t' = \mathbf{0.349375}$$

…and finally, perform spline interpolation!

Seg. (c)	*t* value	Arc length	Sample
0	0.0	0.0	$s_0 = p_0 = (1.0, 1.0)$
0	0.25	0.1059	$s_1 = (1.5, 1.0)$
0	0.5	0.2124	$s_2 = (2.0, 0.95)$
0	0.75	0.3189	$s_3 = (2.5, 0.9)$
0	1.0	0.4254	$s_4 = p_1 = (3.0, 0.85)$
1	0.0	0.4254	$s_4 = p_1 = (3.0, 0.85)$
1	0.25	0.4795	$s_5 = (3.25, 0.9)$
1	0.5	0.5365	$s_6 = (3.5, 1.0)$
1	0.75	0.5983	$s_7 = (3.75, 1.15)$
1	1.0	0.6554	$s_8 = (4.0, 1.25)$
2	0.0	0.6554	$s_8 = (4.0, 1.25)$
2	**0.25**	0.6872	$s_9 = (4.15, 1.25)$
2	**0.5**	0.7194	$s_{10} = (4.3, 1.225)$
2	0.75	0.7516	$s_{11} = (4.45, 1.2)$
2	1.0	0.7839	$s_{12} = (4.6, 1.175)$
3	0.0	0.7839	$s_{12} = (4.6, 1.175)$
3	0.25	0.8368	$s_{13} = (4.85, 1.175)$
3	0.5	0.8899	$s_{14} = (5.1, 1.1875)$
3	0.75	0.9429	$s_{15} = (5.35, 1.2)$
3	1.0	1.0	$s_{16} = (5.6, 1.3)$

Speed Control on Curves

- ***Method 2: Spline***
- *Samples* are no longer relevant, we use spline interpolation on the curve, using keyframes!
- We used Catmull-Rom for sampling, so we must use the same to find the value

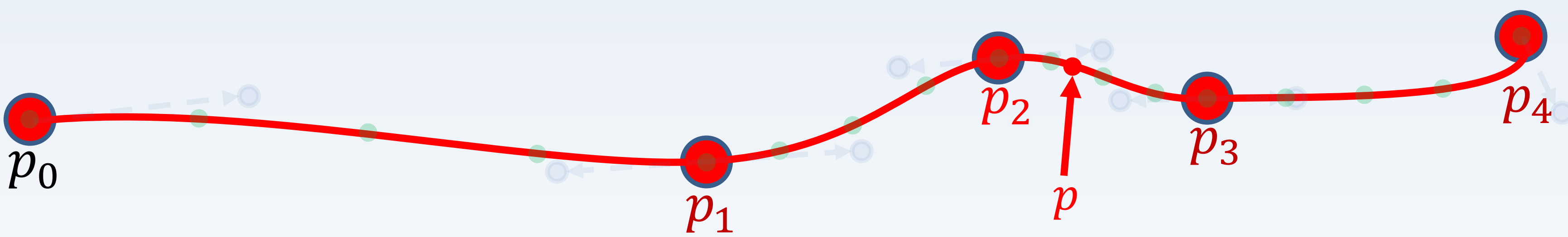


Catmull-Rom interpolation parameter:
$t' = \mathbf{0.349375}$ on *segment 2*
(between keys p_2 and p_3)

Speed Control on Curves

- ***Method 2: Spline***
- Catmull-Rom interpolation, $c = 2$

$$p = \mathrm{CatmullRom}_{p_{c-1}p_cp_{c+1}p_{c+2}}(t')$$
$$= \mathrm{CatmullRom}_{p_1p_2p_3p_4}(0.349375)$$



Catmull-Rom interpolation parameter:
$t' = \mathbf{0.349375}$ on *segment 2*
(between keys p_2 and p_3)

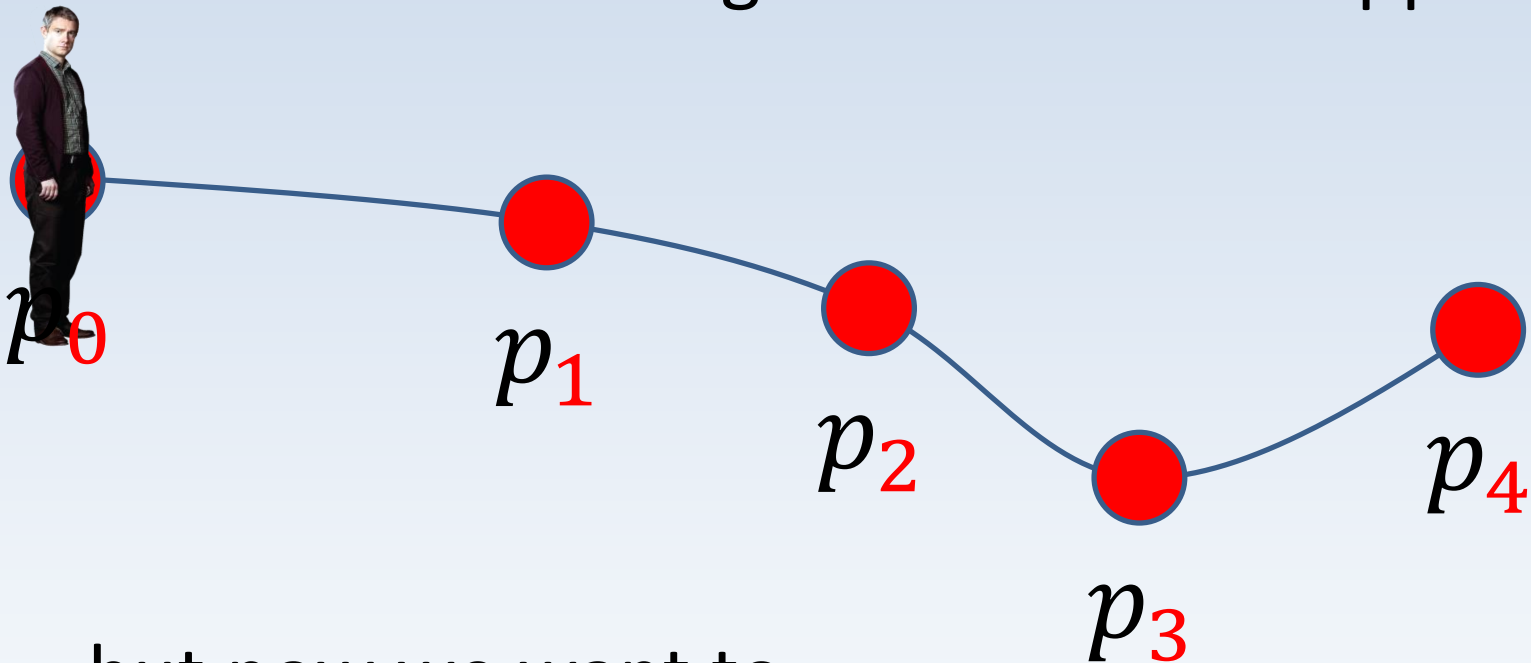
Speed Control on Curves

- With speed control, we no longer interpolate using the raw *t* value we are familiar with…
- The new control value: *normalized arc length*
- We use one of the aforementioned methods (LERP or spline interpolation) to either

1. Approximate (LERP) based on arc length, or
2. Compute a valid interpolation parameter for spline interpolation

Speed Control on Curves

- We have done enough to make this happen:



- …but now we want to actually ***control the speed on the curve***

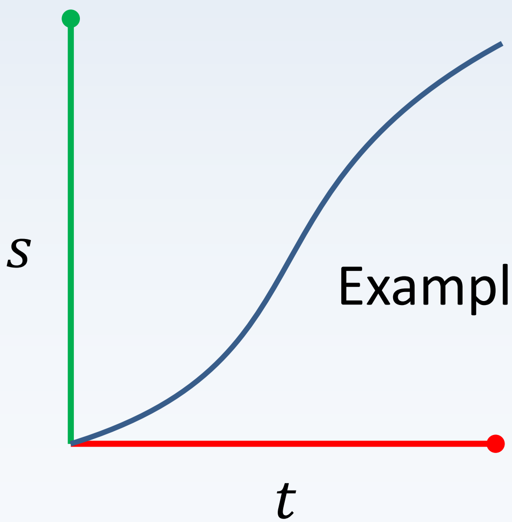
Speed Control on Curves

- Interpolating along the curve using *normalized arc length* as our control parameter yields…???
- …a constant speed along the curve
- This is where we employ measures to *control* the arc length parameter input before it is used for interpolation!

Speed Control on Curves

- *Speed profile*: a function of *distance* with respect to *time* so we can control *speed*:

$s(t)$ = distance (arc length)
function w.r.t. *time*



Remember that *speed* is distance / time

Example speed profile

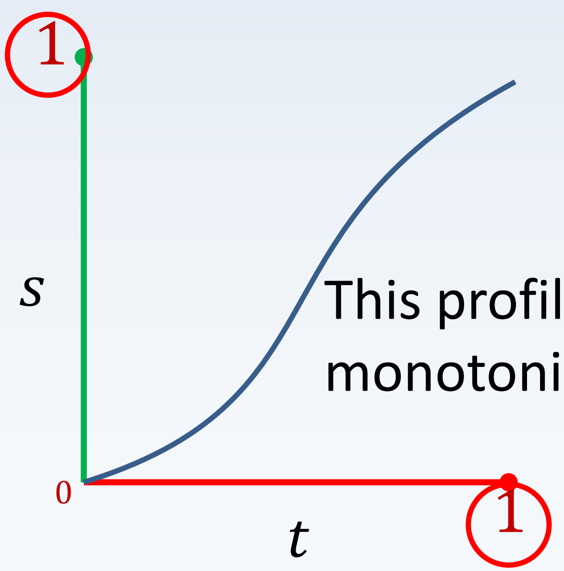
t = time (independent variable)

Speed Control on Curves

- The speed profile works best if the function meets three guidelines:

1. Function is *monotonic*
2. Function is *continuous*
3. Input (t) and output (s) are both *normalized*… why???

$s(t)$ = distance



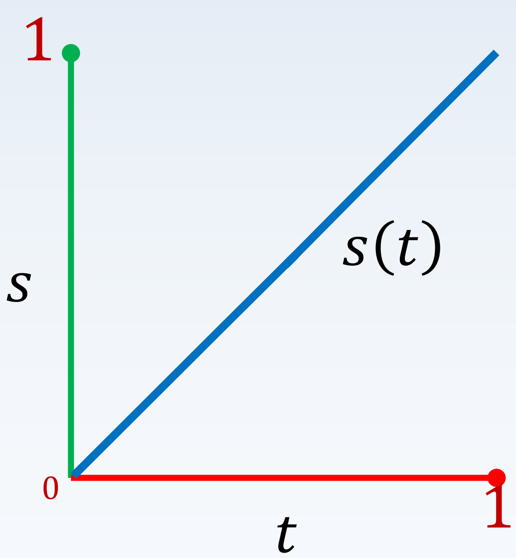
This profile function is both monotonic and continuous.

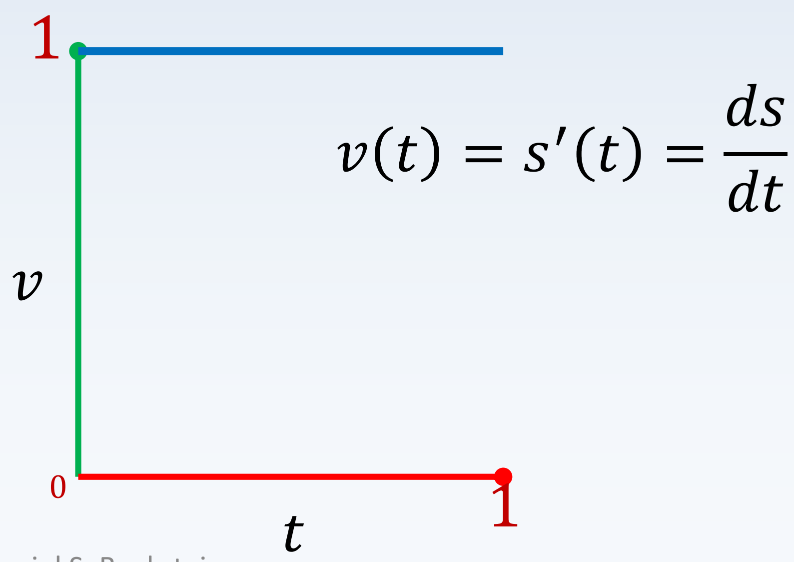
The *t* and *s* axes are both normalized!

t = time

Speed Control on Curves

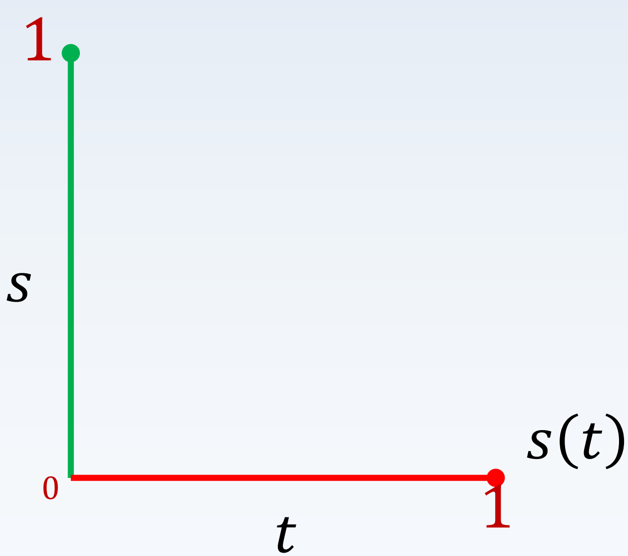
- Example speed profile function: *uniform*
- Distance (arc length) ***is*** the time parameter!
- What does the *speed* function look like then?

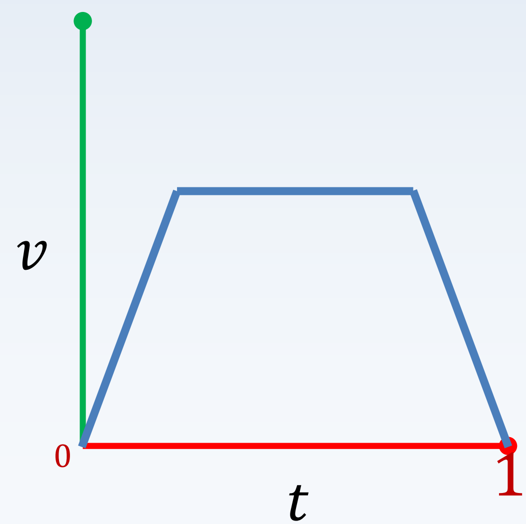




Speed Control on Curves

- What if we want to do some *easing*?
- …there's a function for that!
- Create a *ramp* in the velocity function:

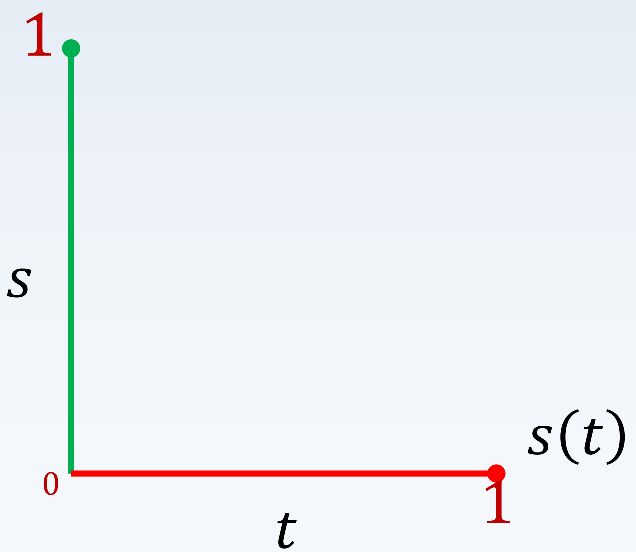


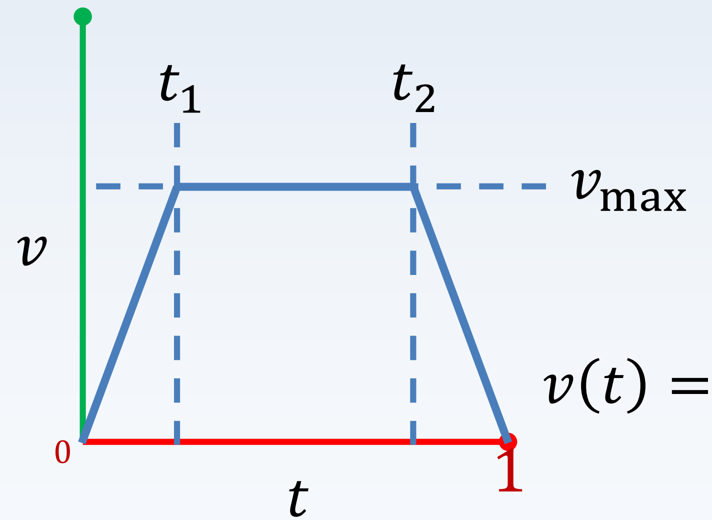


$$v(t) = s'(t) = \frac{ds}{dt}$$

Speed Control on Curves

- We just need to specify the *time values* where the ease-in ends and the ease-out begins:
- The system takes care of the rest: determining the max velocity and how it affects distance!





$$v(t) = s'(t) = \frac{ds}{dt}$$

Speed Control on Curves

- "*Parabolic easing*"
- System computes maximum velocity and the distance curve associated with the velocity

t_1 : ease-in ends

t_2 : ease-out begins

$v_{\max}$: maximum velocity

$v(t)$: velocity at time t

Speed Control on Curves

- "*Parabolic easing*": velocity function

$$v_{\max} = \frac{2}{1 + t_2 - t_1}$$

$$v(t) = \begin{cases} v_{\max}\left(\dfrac{t}{t_1}\right) & \text{if} \quad t < t_1 \\ v_{\max} & \text{if} \quad t_1 \le t \le t_2 \\ v_{\max}\left(1 - \dfrac{t - t_2}{1 - t_2}\right) & \text{if} \quad t_2 < t \end{cases}$$

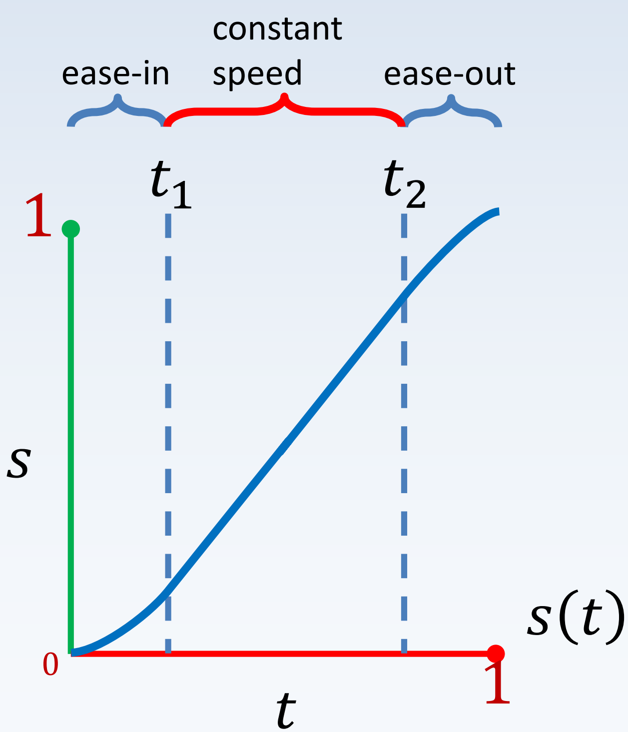
Speed Control on Curves

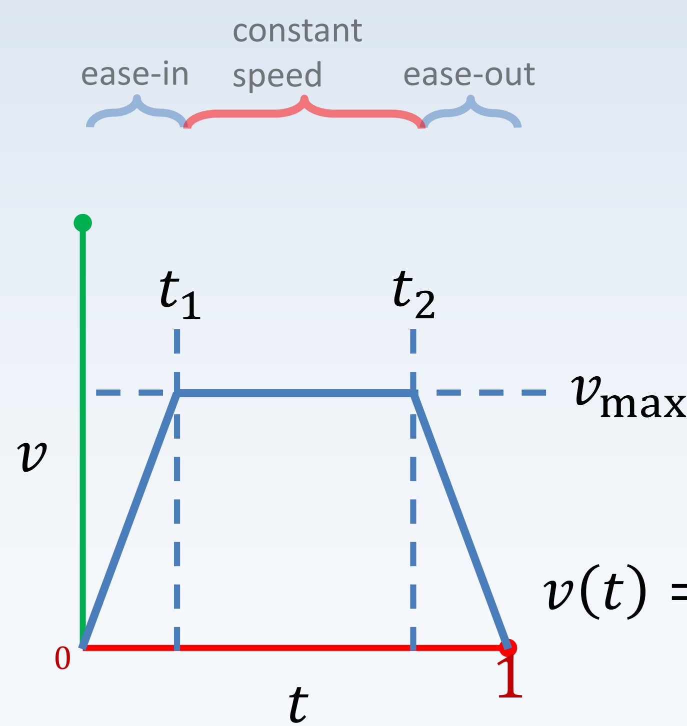
- "*Parabolic easing*": distance function

$$s(t) = \begin{cases} v_{\max}\left(\dfrac{t^2}{2t_1}\right) & \text{if} \quad t < t_1 \\ v_{\max}\left(t - \dfrac{t_1}{2}\right) & \text{if} \quad t_1 \le t \le t_2 \\ v_{\max}\left(t - \dfrac{t_1}{2} - \dfrac{(t - t_2)^2}{2(1 - t_2)}\right) & \text{if} \quad t_2 < t \end{cases}$$

Speed Control on Curves

- The result is a speed profile with easing!
- ‘*s*’ is used to control our spline interpolation!





$$v(t) = s'(t) = \frac{ds}{dt}$$

Speed Control on Curves

- The alternative: go deeper and fiddle around with the *acceleration*
- Integrate acceleration w.r.t. time to get *speed*, integrate speed w.r.t. time to get *distance*
- The hard part is reparametrizing the curve
- After that, you have full control over it by playing around with the speed profile…
- (which is just another *curve*!!!) :o

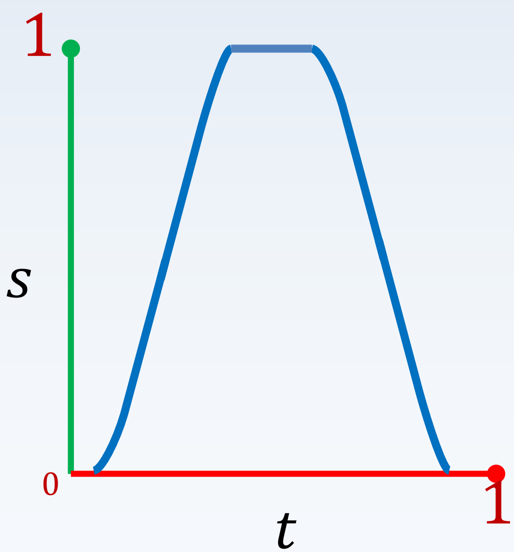
Speed Control on Curves

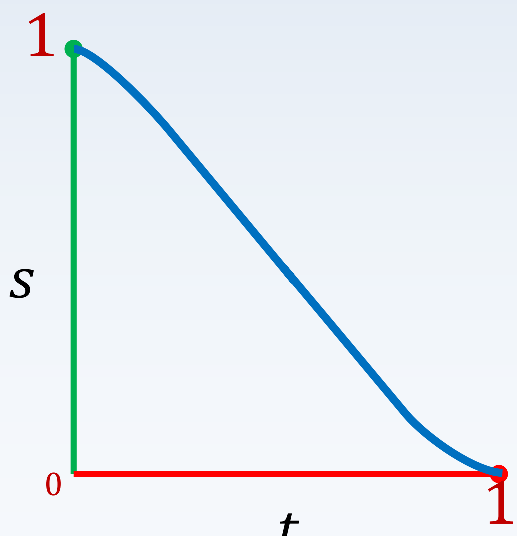
- Never forget: ***speed = distance / time***
- ***If time is constant, then speed is proportional to distance!***
- We control the *distance* along the curve using the *arc length*
- A *speed profile* is a function of distance relative to time… used to convert time into distance, which is used to interpolate!!!

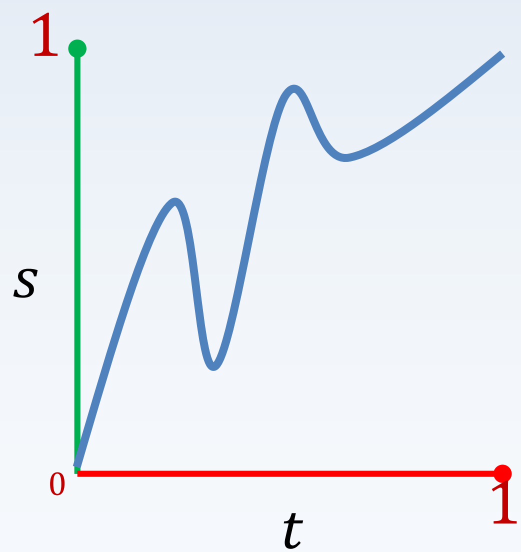
Speed Control on Curves

- Speed profiles are not cut-and-dry…

- What do these ones do???







Speed Control on Curves

- **Speed control summary**:
- Starts with *reparametrizing* the curve
    - Create sample table, use *arc length* to control spline interpolation
    - Result is a constant speed along the curve
- We can create a *speed profile* for the curve
    - Determine how *distance* is affected *over time*
- Use the speed profile curve to control spline interpolation because it represents *arc length*

The end.

- Questions? Comments? Concerns?