Pragmatic Issues in Curves

Drawing
- have algorithm for drawing lines (and hardware)
  - Can we just draw a line?
    - Yes IF small or straight
  - Idea: break curve into pieces
    - \( f(1) \), \( f(2) \), \( f(i) \)

Fixed vs. Adaptive
- pick \( N \) and divide uniformly
- vs.
  - break into pieces, keep breaking until OK

Is it ok?

How do we know how big/curvy a curve segment is:
- Bounding box — would be nice
- Easy to divide to have curve that
- Easy to know ends // do this
- Easy to assess curviness → Bezier's (later)
Simple idea:
try \( f(0.5) \)
if \( f(0.5) \approx \frac{f(0)+f(1)}{2} \)
then line is OK

Note: work for straight or small

**ARC LENGTH Parameterization**

1. if the curve is straight, easy!

   \[
   \frac{u}{1} = \frac{s}{d = \|A-B\|}
   \]

2. given \( u \), getting \( s \) is easy

   break curve into small pieces

   going backwards is harder

   build a table, use lookup / interpolation

<table>
<thead>
<tr>
<th>( u )</th>
<th>( s )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0.1</td>
<td>( s_1 )</td>
</tr>
<tr>
<td>0.2</td>
<td>( s_2 )</td>
</tr>
<tr>
<td>0.3</td>
<td>( s_3 )</td>
</tr>
</tbody>
</table>

   if have \( s \) here, \( s_3 - s_2 \)
   then \( u = 0.2 + \frac{S - s_2}{s_3 - s_2} \cdot (0.3 - 0.2) \)
A TRICK FOR TRAINS

Have \( u \) and \( \Delta s \) (how far between ties, train speed)
compute \( \Delta u \)

Idea: pick a small \( \Delta u \) so straight line is OK
figure out the corresponding \( \Delta s \)
too small? use it as a step
too big? either divide the step or interpolate

\[ \Delta u_1 \quad \Delta u_2 \quad \Delta u_3 \quad \Delta u_4 \quad \Delta s \ (goal) \]

\( \Delta s_1 \quad \Delta s_2 \quad \Delta s_3 \)

how do you know if \( \Delta u_i \) is too big?

\[ \|AB\| \approx \|AA'\| + \|A'B\| \]
then it's nearly straight

Note all this worked for any curve
just evaluate \( f \)

In practice: be careful to get wrap around correct!