Notes on the Unicycle Puzzle (PoW 1248)

Stan Wagon, wagon@macalester.edu, Sept. 2017.

The basic idea here is due to David Finn. Here are papers on the subject.

D. Finn, Can a bicycle create a unicycle track, College Math. Journal, 33:4 (Sept. 2002) 283-292.
https://www.maa.org/sites/default/files/pdf/upload_library/22/Polya/Finn.pdf

R. L. Foote, M. Levi, S. Tabachnikov, Tractrices, bicycle tire tracks, hatchet planimeters, and a 100-year-old conjecture, Amer. Math. Monthly 120:3 (2013) 199-216.
http://www.math.psu.edu/tabachni/prints/FLT8.pdf

Mark Levi and Serge Tabachnikov, Hatchet Planimeter, Menzin's Conjecture, and Oscillation of Unicycle Tracks, On bicycle tire tracks geometry, hatchet planimeter, Menzin's conjecture, and oscillation of unicycle tracks, Experimental Mathematics 18:2 (2009) 173-186.
https://projecteuclid.org/euclid.em/1259158427

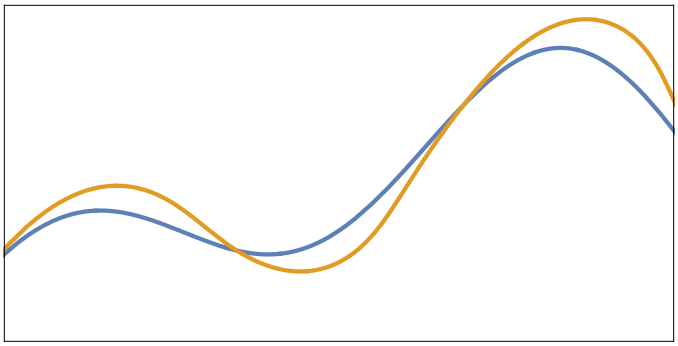
Gil Bor, Mark Levi, Ron Perline, and Sergei Tabachnikov, Tire tracks and integrable curve evolution, preprint, July 2017, http://www.math.psu.edu/tabachni/prints/main.pdf

All code here is *Mathematica*. The `FrontTrack` function gets the front track from the rear using a unit tangent vector.

```
unitVec[v_] := v/Sqrt[v.v];
FrontTrack[rear_, t_] := rear + unitVec[D[rear, t]];
```

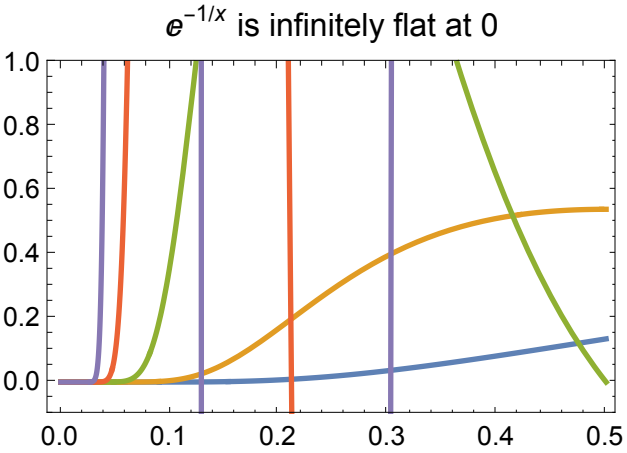
An example: this is useful for generating tracks for the Which Way Did the Bicycle Go? puzzle (as in my book of the same name).

```
rear = {t, Sin[t] + t / 10 - Cos[t / 2]};
ParametricPlot[Evaluate[{rear, FrontTrack[rear, t]}],
 {t, 0, 3 π}, Frame → True, Axes → False,
 FrameTicks → None, PlotRange → {{1, 9}, {-1, 3}}]
```



Reminder: All derivatives of $e^{\frac{-1}{x}}$ are 0 at the origin. This is a nonanalytic function.

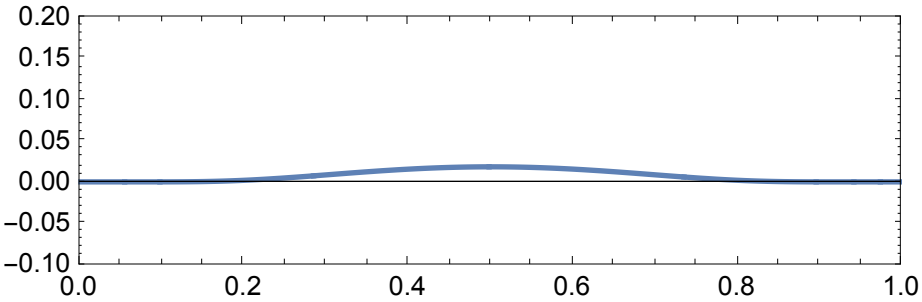
```
Plot[Evaluate[Table[D[e^-1/x, {x, i}], {i, 0, 4}]],
 {x, 0, 0.5}, Frame → True, PlotRange → {-0.1, 1}, Axes → False,
 PlotStyle → Thick, PlotLabel → "e^-1/x is infinitely flat at 0"]
```



David Finn used $\left(t, e^{-\frac{1}{t(1-t)}}\right)$ as the seed curve for the unicycle problem. This parametric curve is infinitely flat at $(0, 0)$ and $(1, 0)$ (and so is nonanalytic). This curve will be used as the path of the rear wheel as it leave the origin.

```
finn[1][t_] := {t, e^-1/t (1-t)};
finn[1][0 | 0.] = {0, 0};
finn[1][1 | 1.] = {1, 0};
```

```
ParametricPlot[finn[1][t], {t, 0, 1},
 PlotRange → {{0, 1}, {-.1, .2}}, PlotStyle → Thick, Frame → True]
```



The front path is determined by the rear, so next we store six iterations of the forward-moving track. This takes a while because of the last one, which has six million terms.

```
Do[finn[i][t_] := Evaluate[FrontTrack[finn[i - 1][t], t]];
   finn[i][0 | 0.] = {i - 1, 0};
   finn[i][1 | 1.] = {i, 0}, {i, 2, 6}]; // AbsoluteTiming
```

```
{1.60871, Null}
```

Here are the number of occurrences of the power function, and also the memory used, for each of these symbolic forms.

```
Table[Count[finn[i][t], _^, ∞], {i, 6}]
Table[ByteCount [finn[i][t]], {i, 6}]
```

```
{3, 28, 385, 6844, 135 496, 2 852 980}
```

```
{480, 4856, 68 872, 1 232 136, 24 466 040, 516 001 000}
```

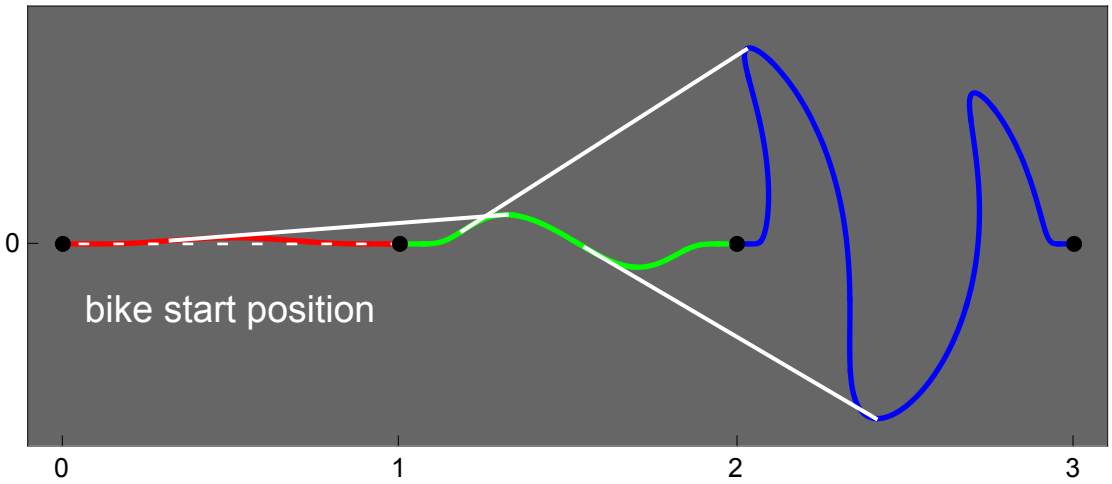
Next we store plots of the first six functions. This takes over one minute.

```
cols = {Red, Green, Blue, Orange, Cyan, Yellow};
nmax = 6;
Do[plot[i] = ParametricPlot[Evaluate[finn[i][t]], {t, 0, 1},
      PlotStyle → {AbsoluteThickness[2], cols[[i]]}, PlotPoints → 30],
    {i, nmax}] // Quiet; // AbsoluteTiming
```

```
{76.5357, Null}
```

The next graphic shows the first three iterations, with a couple of tangent lines from rear to front in white.

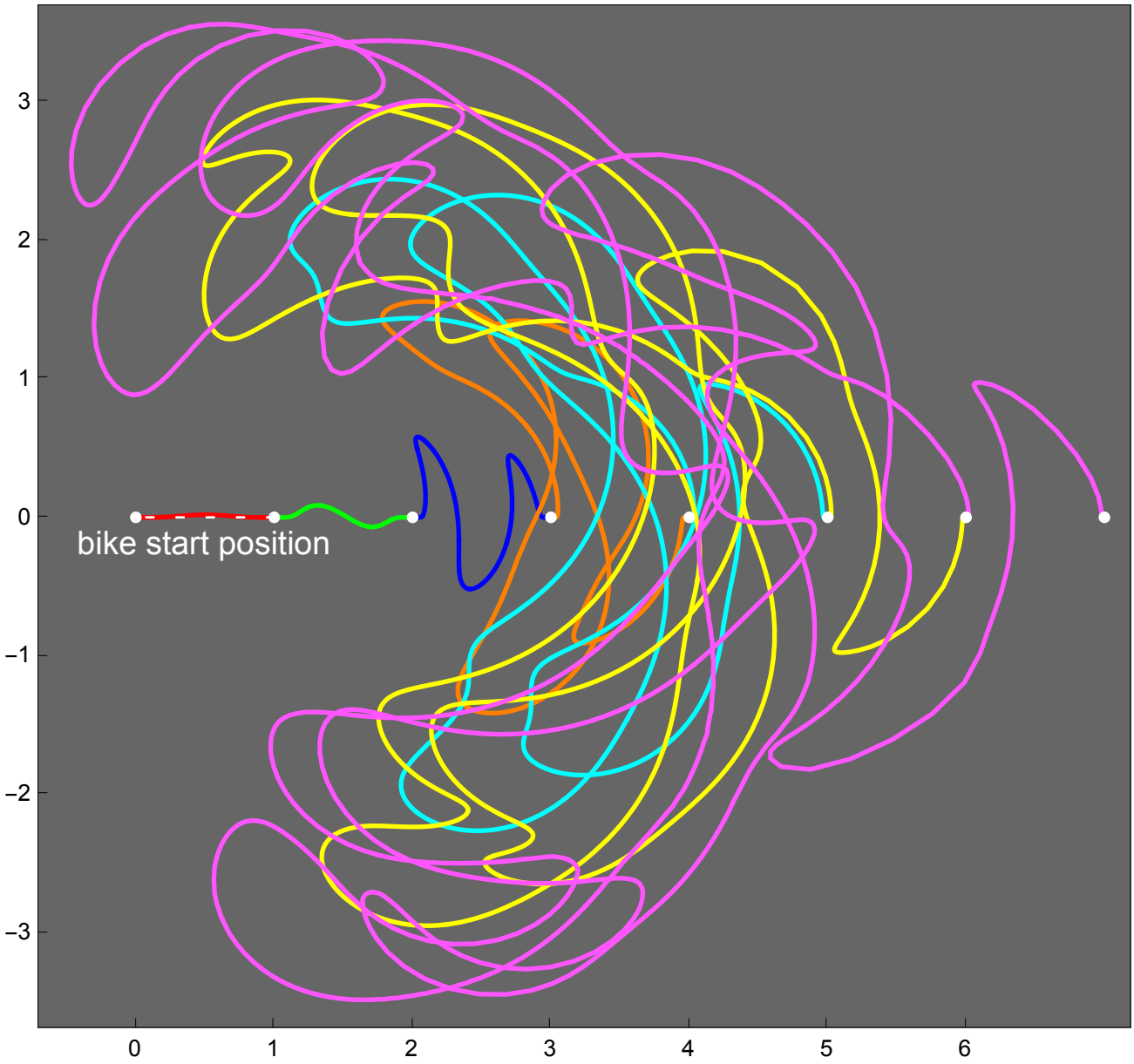
```
nmax = 3;
xmin = -0.1; xmax = nmax + 0.1; ymin = -0.6; ymax = 0.7;
Show[Graphics[{GrayLevel[.4], Rectangle[{xmin, ymin}, {xmax, ymax}],
   Black, White, Text[Style["bike start position", 14], {.5, -.2}]}],
 Table[plot[i], {i, nmax}], Graphics[{{White, AbsoluteDashing[{3, 10}],
    AbsoluteThickness[1], Line[{{1, 0}, {0, 0}}]}, AbsolutePointSize[6],
   Point[Table[{i, 0}, {i, 0, nmax + 1}]], {White, AbsoluteThickness[1.5],
    Table[Line[{finn[1][t], finn[2][t]}], {t, {0.32}}],
    Table[Line[Re[{finn[2][t], finn[3][t]}]], {t, {.185, 0.55}}]}}],
 Frame → True, Axes → None, PlotRange → {{xmin, xmax}, {ymin, ymax}},
 FrameTicks → {Range[0, nmax], Range[-4, 4], None, None},
 ImageSize → 500]
```



The seventh one is tough, so we get all the points used in the sixth plot and use those points to estimate the derivative. I also did a little smoothing on the magenta curve.

```
pts6 = First[Cases[Normal[plot[6]], Line[x_] :> x, ∞]];
Length[pts6]
nmax = 6; xmin = -0.7; xmax = nmax + 1.2; ymin = -3.7; ymax = 3.7;
imageForward = Show[
  Graphics[{GrayLevel[0.4], Rectangle[{xmin, ymin}, {xmax, ymax}], Black,
    White, Text[Style["bike start position", 14], {.5, -.2}]}],
  Table[plot[i], {i, nmax}], Graphics[{{White, AbsoluteDashing[{3, 10}],
     AbsoluteThickness[1], Line[{{1, 0}, {0, 0}}]}}],
  Graphics[{AbsoluteThickness[2], Lighter[Magenta], tt = Table[
      pts6[[i]] + (pts6[[i + 1]] - pts6[[i - 1]])/Norm[pts6[[i + 1]] - pts6[[i - 1]]], {i, 2, Length[pts6] - 1}];
    Line[Join[tt[[{1, 2}]], Table[Mean[tt[[{i - 2, i - 1, i, i + 1, i + 2}]]],
       {i, 3, Length[tt] - 2}], tt[[{-2, -1}]]]], AbsolutePointSize[5],
    White, Point[Table[{i, 0}, {i, 0, nmax + 1}]]}],
  Frame → True, Axes → None, PlotRange → {{xmin, xmax}, {ymin, ymax}},
  FrameTicks → {Range[0, nmax], Range[-4, 4], None, None},
  ImageSize → 800, PlotRangePadding → None]
```

```
1515
```



It appears that further iterations will lead to crossings of the negative x-axis and, perhaps, the full curve to infinity will cross the negative x-axis infinitely many times and arbitrary far out.

One can also generate the rear track from any front track, but that involves numerically solving a differential equation. If the front path is given by $\boldsymbol{f}(t)$, there is an elegant differential equation for the back path, $\boldsymbol{b}(t)$. The derivation rests on the basic rear-front relationship (where L is the bike length, which we take to be 1):

$$\boldsymbol{f}(t) = \boldsymbol{b}(t) \pm L \, \frac{\boldsymbol{b}'(t)}{\|\boldsymbol{b}'(t)\|}.$$

Now, the speed of the back point is given as follows.

$$\|\boldsymbol{b}'(t)\| = \pm \tfrac{1}{L} \, \boldsymbol{f}'(t) \cdot (\boldsymbol{f}(t) - \boldsymbol{b}(t)),$$

where the sign is the same as in the basic relationship and the dot stands for a dot product (the physical reason for this is that the velocity of the back tire equals the magnitude of the

component of the front-tire velocity in the direction of the back-tire motion; mathematically, the second equation can be obtained by differentiating the first one and using the Frenet formulas). Combining the previous two equations yields

$$\boldsymbol{b}'(t) = \frac{1}{L^2}\,(\boldsymbol{f}'(t)\cdot(\boldsymbol{b}(t)-\boldsymbol{f}(t)))\,(\boldsymbol{b}(t)-\boldsymbol{f}(t)).$$

Start with the same bump -- the red curve above -- but work backwards. This code is a little complicated because `NDSolve` requires that the functions mentioned have a certain form.

```
Clear[front1, frontDer, t, tt, front, backSol, backSolDer];
front[t_] := finn[1][t]
front1[0 | 0.] := front[0]
front1[1 | 1.] := front[1]
front1[t_?NumericQ] := front[t];
frontDer[0 | 0. | 1 | 1.] := {1, 0}
frontDer[t_?NumericQ] := Evaluate[D[front[t], t]];
backSol[0] = front1;
backSolDer[0] = frontDer;
backSol[1] = b /. NDSolve[
    {b'[t] == backSolDer[0][t].(b[t] - backSol[0][t]) (b[t] - backSol[0][t]),
     b[1] == backSol[0][0]}, b, {t, 0, 1}][[1]]
backSolDer[1][t_?NumericQ] := Evaluate[ D[backSol[1][tt], tt] /. tt → t]
```

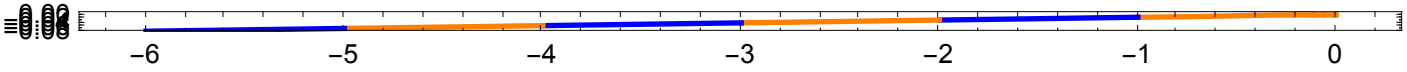
```
InterpolatingFunction[ 
    Domain: {{0., 1.}}
    Output dimensions: {2} ]
```

The preceding gets the first backward step as an InterpolatingFunction. Now we can proceed in automated faction to move farther back.

```
Do[backSol[i] = b /. First[NDSolve[
     {b'[t] == backSolDer[i - 1][t + 1].(b[t] - backSol[i - 1][t + 1])
        (b[t] - backSol[i - 1][t + 1]),
      b[-i + 2] == backSol[i - 1][-i + 2]}, b, {t, -i + 1, -i + 2}]];
 backSolDer[i][t_?NumericQ] := Evaluate[
   D[backSol[i][tt], tt] /. tt → t], {i, 2, 6}]
```

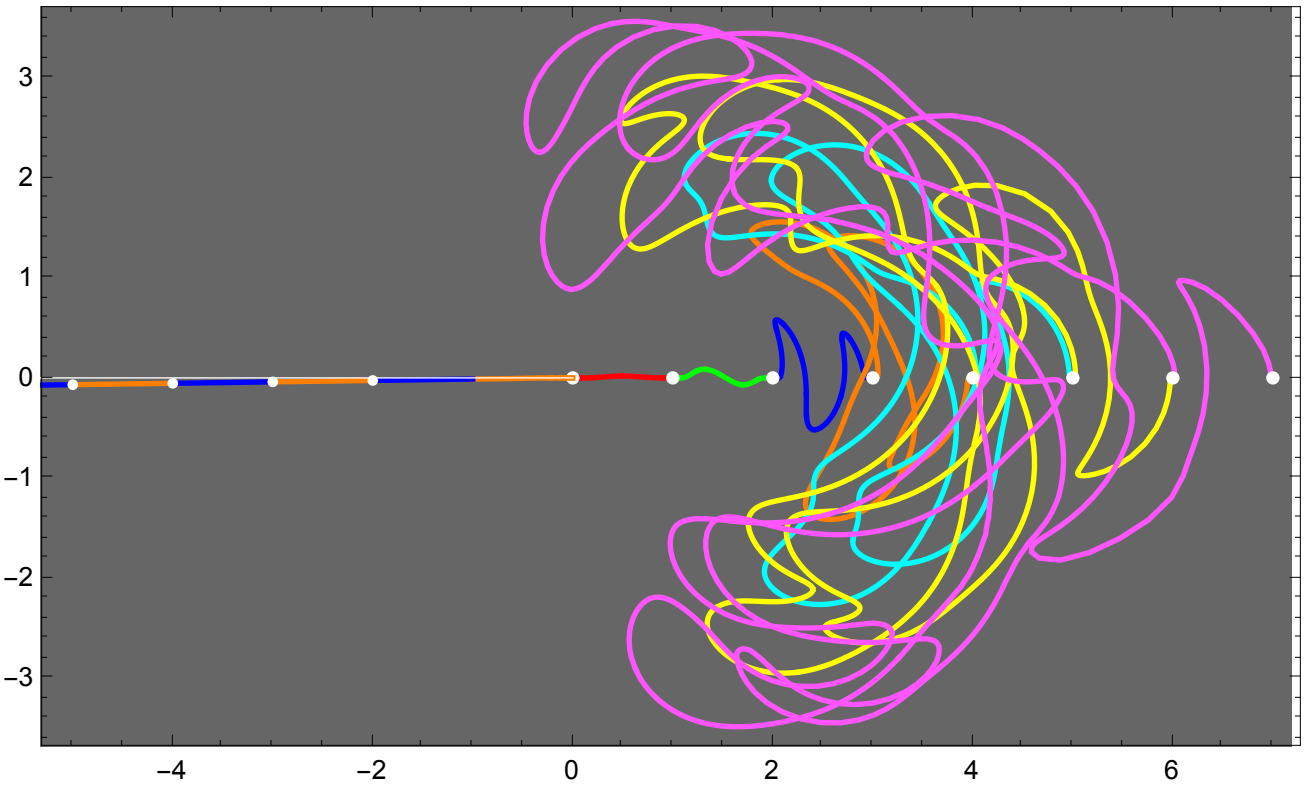
The resulting path is very close to linear and has a slope near 0.014.

```
imageBack = Show[
  Table[ParametricPlot[Evaluate[backSol[i][t]], {t, -i + 1, -i + 2},
    PlotStyle → {If[EvenQ[i], Blue, Orange], AbsoluteThickness[2]}],
   {i, 1, 6}], PlotRange → All, Frame → True]
```



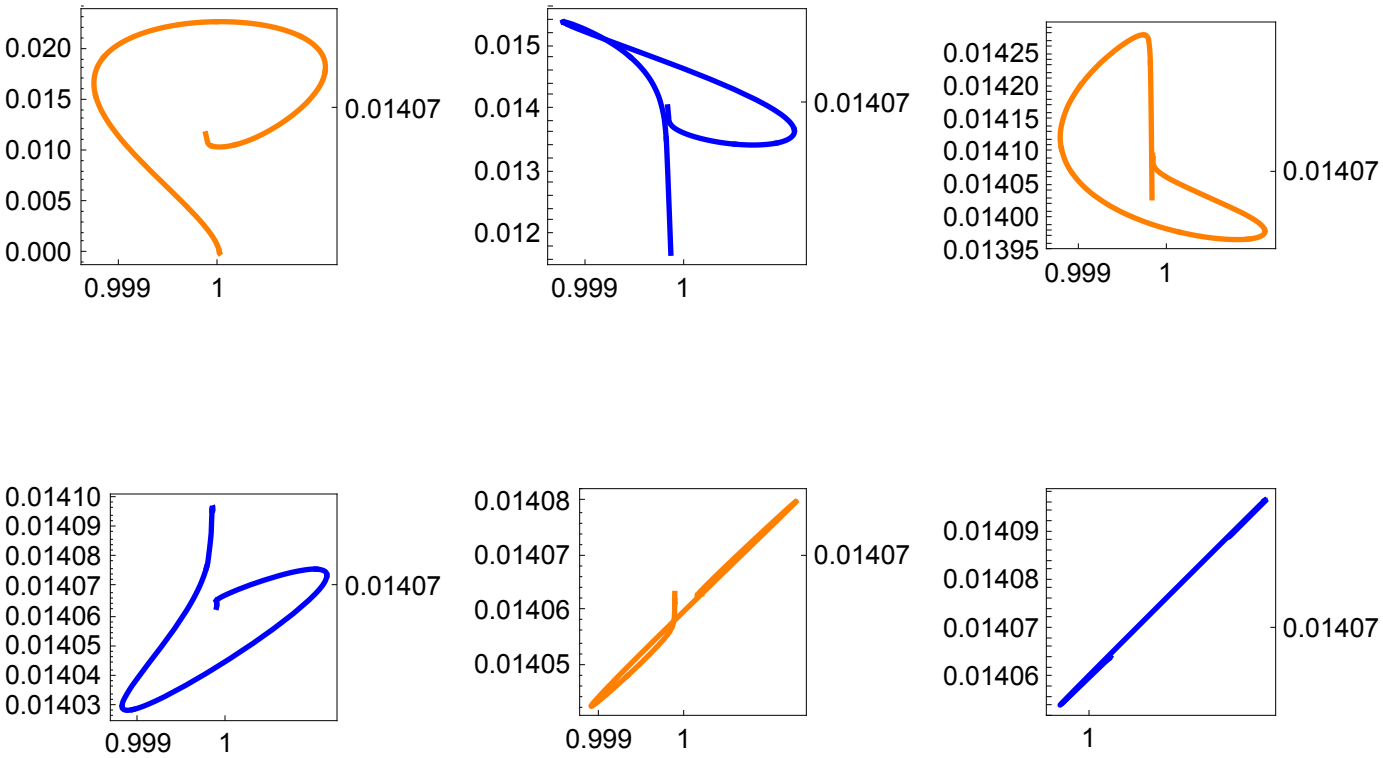
So now we can look at a bidirectional plot of part of a bike=unicycle path that is infinite in both directions.

```
Show[Graphics[{GrayLevel[.4], Rectangle[{-5.5, ymin}, {xmax, ymax}]}],
 DeleteCases[DeleteCases[imageForward, Text[___], ∞],
  {__, AbsoluteDashing[_], __}, ∞], imageBack,
 Graphics[{White, AbsolutePointSize[4], Line[{{-5.5, 0}, {0, 0}}],
   Point[Table[backSol[i][-i + 1], {i, 2, 5}]]}],
 PlotRange → {{-5.3, 7.3}, {ymin, ymax}},
 PlotRangeClipping → True, PlotRangePadding → 0,
 Frame → True, FrameTicks → Automatic, ImageSize → 600]
```



Here are the derivatives of the tracks going backward. The scales decrease and this shows the slope is nearing 0.01407.

```
GraphicsGrid[
 Partition[
  Table[ParametricPlot[Evaluate[D[backSol[i][t], t]], {t, -i + 1, -i + 2},
    PlotStyle → {If[EvenQ[i], Blue, Orange], AbsoluteThickness[2]},
    Frame → True, FrameTicks → {{ Automatic, {0.01407}}, {{1, .999}, {}}},
    AspectRatio → 1, PlotRange → All], {i, 1, 6}], 3], ImageSize → 600]
```



It is an open question whether there is a fake-unicycle track made from an analytic function. Bill Bixler worked on some approximations, and the following track, while not 100% accurate, gives a nice approximate solution. Let

$f_i(t) = \left(\sqrt{i} + \left(\sqrt{i+1} - \sqrt{i}\right) t\right)(\cos[2\pi t], \sin[2\pi t])$. Then plotting f_i for $i \geq 1$ and with $0 \leq t \leq 1$ in each case gives the following, where many tangents are shown. A problem is that this curve is not differentiable where it crosses the x-axis. Perhaps one can use a nonanalytic radius function to get something like this that works for infinitely many loops; I tried a few things without success.