

# Mathematica(l) Roller Coasters

Selwyn Hollis  
Department of Mathematics  
Armstrong Atlantic State University  
Savannah, GA 31419  
shollis@armstrong.edu

## ■ Introduction

Mathematica animations that simulate roller coasters provide an interesting and fun way of putting basic ideas from multivariable calculus and elementary physics to work. In this article we will describe how to create “monorail” roller coaster animations in which the track is a surface in  $\mathbb{R}^3$  in the form of a narrow strip. The space curve comprising the center “rail” of the track may be specified either as a closed “coordinate curve” on a given parametric surface or as a closed space curve. In the first case, the orientation of the vehicle makes use of partial derivatives. In the second case, the orientation of the vehicle makes use of the curve’s classical *TNB frame*, which consists of the tangent, normal, and binormal vectors.

## ■ The Physics

The motion of our simulated roller coaster will be made to appear realistic by our use of the principle of conservation of energy. We will assume a frictionless track and that upward is the positive vertical direction. So at all times we have

$$\Delta(\text{KE} + \text{PE}) = 0,$$

where  $\text{KE} = m v^2/2$  is kinetic energy, and  $\text{PE} = m g z$  is potential energy. Here  $m$  is mass,  $g$  is the gravitational acceleration constant,  $z$  is vertical displacement from any reference point, and  $v$  is speed. Since there is no force—and consequently no change in potential energy—in any horizontal plane, we actually have

$$\text{KE} = m v_z^2/2 + c$$

where  $v_z$  is the vertical component of the velocity and  $c$  is a constant. So from the conservation principle above, it follows easily that

$$\Delta(v_z^2) \propto \Delta z.$$

Suppose that at the highest point on the roller coaster’s track we have  $v_z = v_{\text{top}}$  and  $z = z_{\text{top}}$ . Then at any point on the track,

$$v_z^2 - v_{\text{top}}^2 = k(z_{\text{top}} - z),$$

where  $k > 0$ . The result of all this is that if the position of our roller coaster as a function of time  $t$  is given by  $R(t) = (x(t), y(t), z(t))$ , then

$$x'(t)^2 + y'(t)^2 \text{ is constant, and } z'(t)^2 = v_z^2 = k(z_{\text{top}} - z(t)) + v_{\text{top}}^2.$$

## ■ Determining Position

The center rail of the track will be a closed space curve with a given parametrization

$$r(u) = (r_1(u), r_2(u), r_3(u)).$$

This parameterization will only describe the *path* of the vehicle, not its position as a function of time. Since an animation will consist of a sequence of frames that show the roller coaster at equally-spaced times

$$0, \Delta t, 2 \Delta t, 3 \Delta t, \dots,$$

we need to determine the position of the vehicle at each of these times. This will be done by computing a corresponding sequence of values of the parameter  $u$ , which will entail recomputing the parameter step  $\Delta u$  for each successive frame. The method for computing each successive  $\Delta u$  is developed as follows.

Suppose that the actual position of the vehicle at time  $t$  is given by  $R(t) = (x(t), y(t), z(t))$ . Then, thinking of  $u$  as a function of  $t$ , we have

$$R(t) = r(u(t)),$$

from which follows

$$R'(t) = \frac{dr}{du} \frac{du}{dt}$$

by the Chain Rule. We assume that  $\frac{du}{dt} > 0$ , so it follows that

$$\frac{du}{dt} = \|R'(t)\| / \left\| \frac{dr}{du} \right\|.$$

Now, since  $x'(t)^2 + y'(t)^2$  is constant and  $z'(t)^2 = v_z^2 = k(z_{\text{top}} - z(t)) + v_{\text{top}}^2$ , this becomes

$$\frac{du}{dt} = \sqrt{k(z_{\text{top}} - z(t)) + c} / \left\| \frac{dr}{du} \right\|,$$

where  $c$  is a constant. Thus, the variable parameter step corresponding to the fixed time step  $\Delta t$  can be estimated as

$$\Delta u \approx \frac{\sqrt{k(z_{\text{top}} - r_3(u)) + c}}{\|r'(u)\|} \Delta t,$$

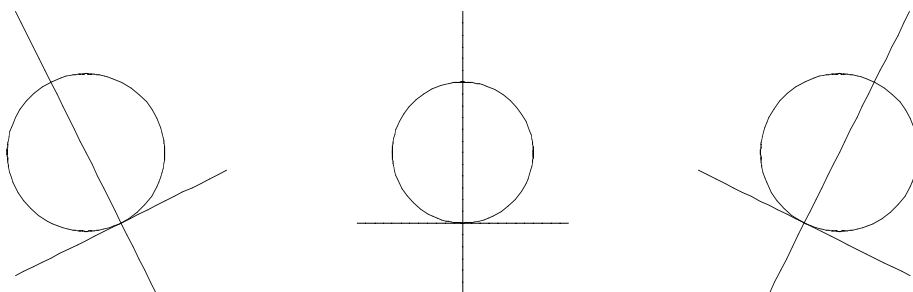
which can be computed without knowledge of  $R(t)$ . Since  $\Delta t$  is fixed, we will introduce new constants  $\tilde{k}$  and  $\varepsilon$  and rewrite this simply as

$$\Delta u \approx \tilde{k} \frac{\sqrt{z_{\text{top}} - r_3(u) + \varepsilon}}{\|r'(u)\|}.$$

The apparent speed of the vehicle in our animations will be controlled by adjusting the constants  $\tilde{k}$  and  $\varepsilon$ .

## ■ The Vehicle and Its Orientation

The vehicle will be represented by a flexible tube connected to the center rail of the track. Its cross-sections orthogonal to the track will be circles that intersect the track's center rail. At each point on the center rail there is a plane that is orthogonal to the path at that point and contains a circular cross-section of the tube. In that plane, imagine two coordinate axes, a "horizontal" axis that is parallel to the track surface and a "vertical" axis that is orthogonal to the track. These axes are indeed horizontal and vertical with respect to the orientation of the vehicle.



So in order to keep the vehicle oriented properly at each location along the track, we need to compute a pair of orthogonal vectors that point in the directions of the axes just described. These vectors are:

- (i)  $N(u)$ , a unit vector that is orthogonal to the center rail and parallel to the track surface;
- (ii)  $B(u)$ , a unit vector that is orthogonal to both the center rail and  $N(u)$ .

With these vectors defined, we can now describe the surface we will use to represent the vehicle at any point  $r(u)$  on the track. Its parametrization is

$$\Phi(s, \theta) = r(s) + \rho B(s) (1 + \cos \theta) + \rho \sin \theta N(s), \quad u - \ell/2 \leq s \leq u + \ell/2, \quad 0 \leq \theta \leq 2\pi.$$

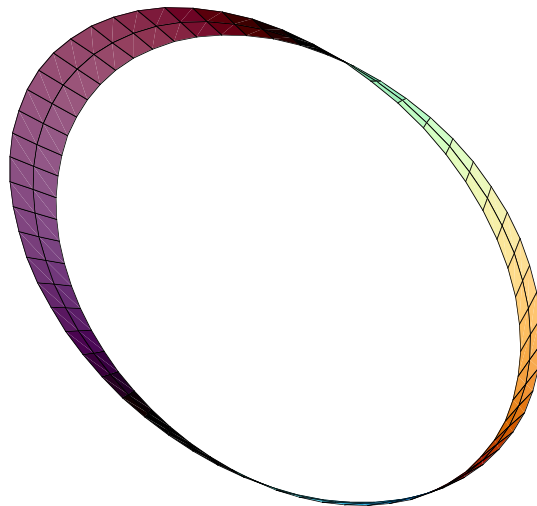
where  $\rho$  is the radius of the vehicle's circular cross-sections, and  $\ell$  is the vehicle's length.

## ■ A Track Given by a Closed Coordinate Curve on a Parametric Surface

A nice example is provided by a Möbius strip, an example of which is parametrized and plotted as follows.

$$\mathbf{q}[\mathbf{u}_-, \mathbf{v}_-] := \{ (2 - \mathbf{v} \sin[\mathbf{u} / 2]) \cos[\mathbf{u}], \mathbf{v} \cos[\mathbf{u} / 2], (2 - \mathbf{v} \sin[\mathbf{u} / 2]) \sin[\mathbf{u}] \};$$

```
track = ParametricPlot3D[Evaluate[q[u, v]], {u, 0, 2 π},
  {v, -.2, .2}, PlotPoints -> {60, 3}, Boxed -> False, Axes -> False];
```



## ■ The Vector Computations

The track's center rail is the “coordinate curve” parametrized by

$$\mathbf{r}[\mathbf{u}_] = \mathbf{q}[\mathbf{u}, 0]$$

$$\{2 \cos[u], 0, 2 \sin[u]\}$$

(By “coordinate curve” we mean a curve with parametrization given by setting one of the coordinates of the surface parametrization equal to a constant.)

The corresponding unit tangent vector is

$$\mathbf{unitTan}[\mathbf{u}_] = \frac{\mathbf{r}'[\mathbf{u}]}{\sqrt{\mathbf{r}'[\mathbf{u}] \cdot \mathbf{r}'[\mathbf{u}]}} // \mathbf{Simplify}$$

$$\{-\sin[u], 0, \cos[u]\}$$

A vector that is parallel to the track surface and not parallel to the curve is given by  $\frac{\partial \mathbf{q}}{\partial v}$  evaluated at  $v = 0$ .

$$\mathbf{vec2}[\mathbf{u}_] = \partial_v \mathbf{q}[\mathbf{u}, \mathbf{v}] /. \mathbf{v} \rightarrow 0 // \mathbf{Simplify}$$

$$\{-\cos[u] \sin\left[\frac{u}{2}\right], \cos\left[\frac{u}{2}\right], -\sin\left[\frac{u}{2}\right] \sin[u]\}$$

One step of the Gram-Schmidt process produces a vector that is parallel to the track surface and *orthogonal* to the curve:

$$\mathbf{nrml}[\mathbf{u}_] = \mathbf{vec2}[\mathbf{u}] - \mathbf{vec2}[\mathbf{u}] \cdot \mathbf{unitTan}[\mathbf{u}] \mathbf{unitTan}[\mathbf{u}]$$

$$\{-\cos[u] \sin\left[\frac{u}{2}\right], \cos\left[\frac{u}{2}\right], -\sin\left[\frac{u}{2}\right] \sin[u]\}$$

The corresponding unit vector is

```
unitNrml[u_] = nrml[u] / Sqrt[nrml[u].nrml[u]] // TrigExpand // Simplify
{-Cos[u] Sin[u/2], Cos[u/2], -Sin[u/2] Sin[u]}
```

This is the vector denoted above by  $N(u)$ . (Note that in this example, the last two steps produced no change. These steps are needed in general, however.)

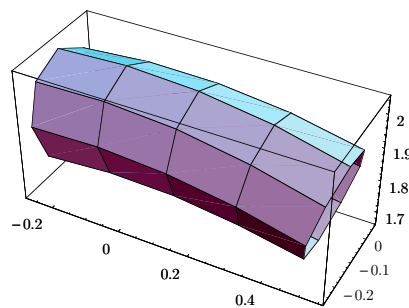
Finally, the cross product  $T(u) \times N(u)$  produces the vector denoted above by  $B(u)$ :

```
biNrml[u_] = unitTan[u] x unitNrml[u] // Simplify
{-Cos[u/2] Cos[u], -Sin[u/2], -Cos[u/2] Sin[u]}
```

## ■ The Vehicle

The following produces the vehicle at any point along the curve. Below that is a sample plot using  $u = 1.5$ .

```
tube[u_] := ParametricPlot3D[Evaluate[
  r[s] + .15 ((1 + Cos[v]) biNrml[s] + Sin[v] unitNrml[s])], {s, u - .2, u + .2},
  {v, 0, 2 Pi}, PlotPoints -> {5, 9}, DisplayFunction -> Identity];
Show[tube[1.5], DisplayFunction -> $DisplayFunction];
```



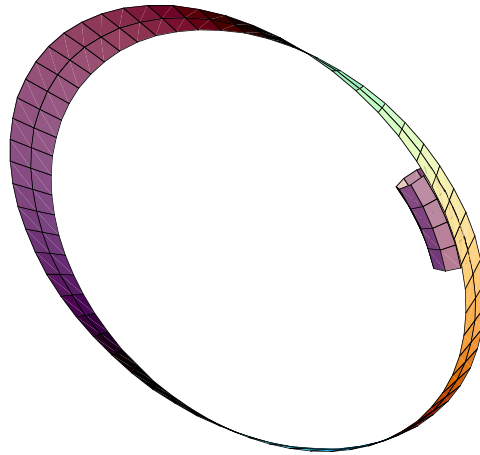
## ■ The Animation

Now we are ready to compute the frames for the animation. The first line below defines the function that computes the parameter step  $\Delta u$ . The numbers .02 and .67 there were chosen experimentally. Forty-eight frames will be produced, the first of which is shown below.

```

ztop = 2;
uStep[u_] = .67  $\frac{\sqrt{ztop - r[u][[3]] + .02}}{\sqrt{r'[u] \cdot r'[u]}}$ ;
u = 0; While[u ≤ 4 π, u = u + uStep[u];
Show[tube[u], track,
  Axes -> False, PlotRange -> {{-2.25, 2.25}, {-1.1, 1.1}, {-2.25, 2.25}},
  Boxed -> False, DisplayFunction -> $DisplayFunction ]]; Clear[u]

```

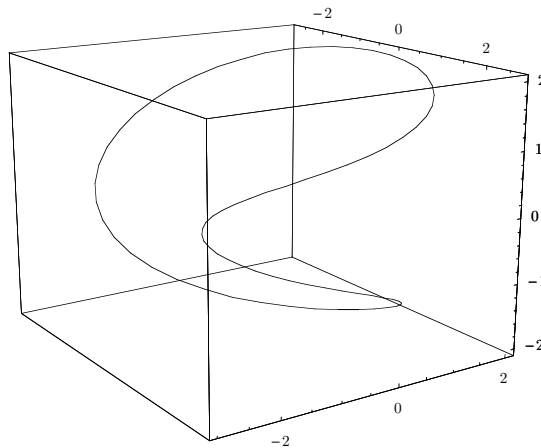


## ■ A Track Given by a Closed Space Curve

An interesting track is obtained from the curve parametrized by

```
r[u_] := {-(2 + Cos[u]) Sin[2 u], (2 + Sin[u]) Cos[2 u], 2 Cos[u]}
```

```
ParametricPlot3D[Evaluate[r[u]], {u, 0, 2 π}, ViewPoint -> {3, -2, 1}];
```



## ■ The Vector Computations

In this situation the vectors of interest comprise the classical *TNB frame* at each point along the curve. Because even fairly simple curves give rise to very large and complicated symbolic expressions for the necessary vectors, we will not calculate them explicitly.

The tangent vector of interest here is

$$\mathbf{tanV}[u\_]=\mathbf{r}'[u]\ //\ \mathbf{TrigExpand}\ //\ \mathbf{Simplify}$$
$$\left\{-\frac{\cos[u]}{2}-4\cos[2u]-\frac{3}{2}\cos[3u],-\frac{\cos[u]}{2}+\frac{3}{2}\cos[3u]-4\sin[2u],-2\sin[u]\right\}$$

The corresponding *unit tangent vector*  $T(u)$  is

$$\mathbf{unitTan}[u\_]:=Module[\{\mathbf{v}=\mathbf{tanV}[u]\},\frac{\mathbf{v}}{\sqrt{\mathbf{v}\cdot\mathbf{v}}}]$$

We will approximate the derivative of this unit tangent vector by a central difference approximation:

$$\mathbf{dT}[u\_]:= (\mathbf{unitTan}[u+.0005]-\mathbf{unitTan}[u-.0005])/.001$$

Now, the *principal unit normal vector*  $N(u)$  is

$$\mathbf{unitNrml}[u\_]:=Module[\{\mathbf{n}=\mathbf{dT}[u]\},\frac{\mathbf{n}}{\sqrt{\mathbf{n}\cdot\mathbf{n}}}]$$

Finally the cross product  $T(u)\times N(u)$  produces the *binormal vector*  $B(u)$ :

$$\mathbf{biNrml}[u\_]:= \mathbf{unitTan}[u]\times\mathbf{unitNrml}[u]$$

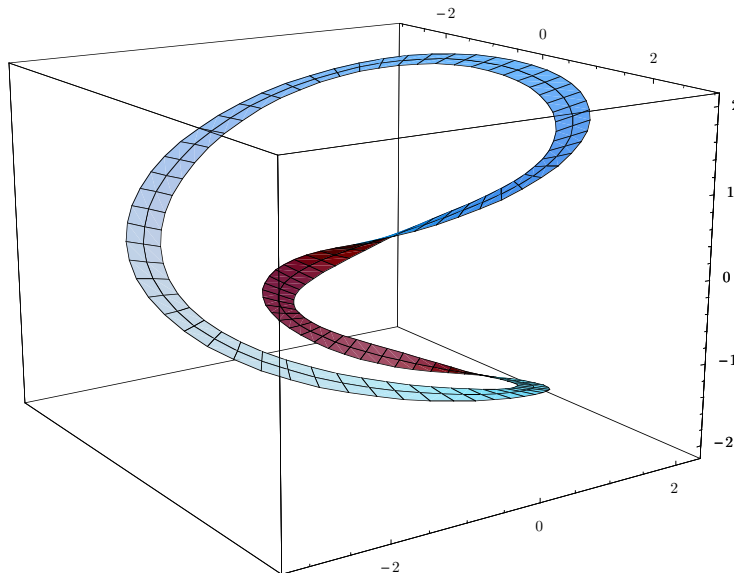
## ■ The Track and the Vehicle

To produce the track from the curve, we use the vector  $N(u)$  to create a surface parametrized by

$$\Gamma(u,v)=r(u)+vN(u),\quad 0\leq u\leq 2\pi,\quad -w/2\leq v\leq w/2,$$

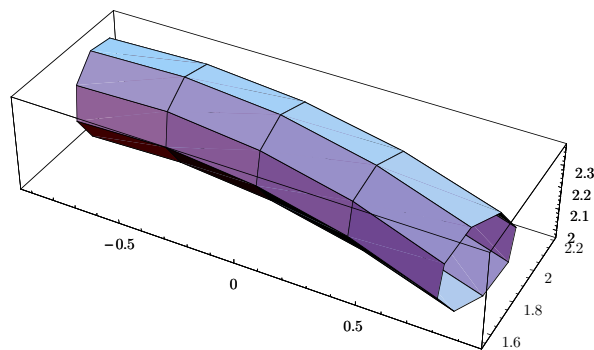
where  $w$  is the width of the track.

```
track = ParametricPlot3D[r[u] + v unitNrml[u], {u, 0, 2 π},
  {v, -.2, .2}, ViewPoint -> {3, -2, 1}, PlotPoints -> {100, 3}];
```



The following produces the vehicle at any point along the curve. Below that is a sample plot using  $u = 0$ . (Note the absence of the Evaluate function inside of ParametricPlot3D here. This is because of the way the unit normal and binormal vectors were defined above with delayed evaluation.)

```
tube[u_] := ParametricPlot3D[
  r[s] + .2 biNrml[s] (1 + Cos[θ]) + .2 Sin[θ] unitNrml[s], {s, u - .15, u + .15},
  {θ, 0, 2 π}, PlotPoints -> {5, 9}, DisplayFunction -> Identity];
Show[tube[0], DisplayFunction -> $DisplayFunction];
```



## ■ The Animation

This animation is created in essentially the same way as the previous one. Forty frames will be produced with the particular parameter values used. The GraphicsArray that follows contains frames 1, 14, 20, and 22.

```
ztop = 2;
```

$$uStep[u_] = \frac{\sqrt{ztop - r[u][[3]] + .02}}{\sqrt{r'[u] \cdot r'[u]}};$$

```
u = 0; While[u ≤ 2 π, u = u + uStep[u];
```

```
Show[tube[u], track, ViewPoint → {3, -2, 1},
```

```
  Axes -> False, PlotRange -> {{-3.6, 3}, {-3.3, 3.3}, {-2.1, 2.4}},
```

```
  DisplayFunction -> $DisplayFunction ]]; Clear[u]
```

