Ball Bouncing on Hilly Terrain

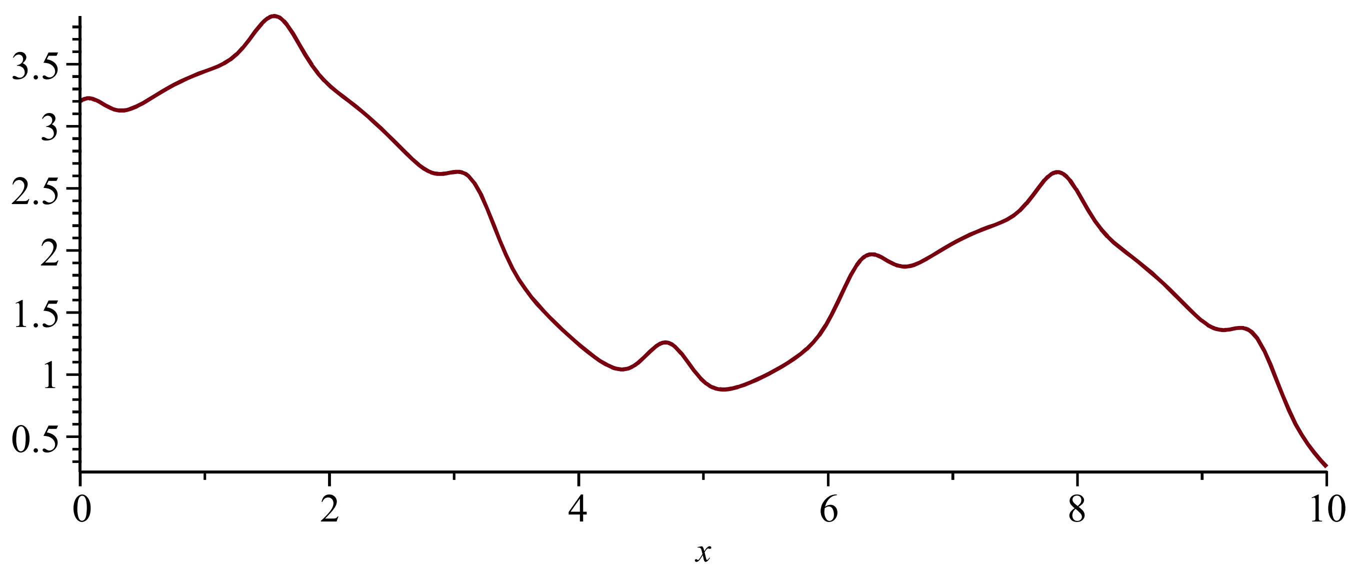
The following application demonstrates how event modeling in the dsolve command can be used to model a ball bouncing on a hilly terrain.

```
> restart;
> with(plots) :
```

Define the Surface

We define the surface, surf, in the line below. It is possible to modify this equation to show different "terrains".

```
> surf := sin(x) + (0.2 cos(4 x + sin(4 x))) − 0.2 x + 3 :
> plot(surf, x = 0 ..10, scaling = constrained)
```



Derive the Hilly Surface

The velocity vector of a ball:

```
> V := Vector([ẋ(t), ẏ(t)]) :
```

The surface normal:

```
> N := LinearAlgebra:-Normalize(Vector([[-(∂/∂x surf)], [1]]), 2, conjugate = false)
      assuming positive :
> N := eval(N, x = x(t))
```

$$N := \begin{bmatrix} \frac{-\cos(x(t)) + 0.2\sin(4x(t) + \sin(4x(t)))(4 + 4\cos(4x(t))) + 0.2}{\sqrt{1 + (-\cos(x(t)) + 0.2\sin(4x(t) + \sin(4x(t)))(4 + 4\cos(4x(t))) + 0.2)^2}} \\ \frac{1}{\sqrt{1 + (-\cos(x(t)) + 0.2\sin(4x(t) + \sin(4x(t)))(4 + 4\cos(4x(t))) + 0.2)^2}} \end{bmatrix}$$

The velocity vector after reflection across the surface normal:

```
> V_reflect := -(1 + C_R) (V.N) N + V :
> V_reflect := simplify(V_reflect, symbolic) :
```

Restitution coefficient:

```
> C_R := 0.99 :
```

Using $C_R < 1$ represents an inelastic collision between the ball and the surface, whereas using $C_R = 1$ represents an elastic collision.

Differential Equations and Initial Conditions

Gravity acts in the -y direction:

```
> deqs := ÿ(t) = -9.81, ẍ(t) = 0 :
> ics := D(x)(0) = 0, D(y)(0) = 0, x(0) = 2, y(0) = 4.5 :
```

Solve and Animate the Differential Equations

```
> sol := dsolve({deqs, ics}, {x(t), y(t)}, numeric,
                events = [[[y(t) = eval(surf, x = x(t)), [temp = ẋ(t), ẋ(t) = V_reflect[1], ẏ(t) = subs(
      ẋ(t) = temp, V_reflect[2])]]],
                range = 0 ..10,
                output = listprocedure) :
```

Animate the Ball Bouncing on the Terrain

```
> xanim := subs(sol, x(t)) :
  yanim := subs(sol, y(t)) :
> p1 := plot(surf, x = 0 ..10, color = black, filled = true, transparency = 0.5) :
  p2 := animate(pointplot, [[xanim(t), yanim(t)], symbol = solidcircle, symbolsize = 15, color
        = black], t = 0 ..10, frames = 150) :
```

> display(p1, p2, view = [0 ..10, 0 ..5], scaling = constrained)

$$t = 0.$$