Exploring the Mathematics Behind Skateboarding: Analysis of the Directional Derivative

Introduction

OBJECTIVE: You will see how the directional derivative changes as you traverse a path through the domain of a function z = f(x,y) .

What is a directional derivative and what does it look like? To find out, you get to put yourself in the position of a skateboarder and explore skating on a flat ramp or inside a bowl. The graphical representation of the dot product of the gradient of the surface function and a unit vector in the direction of motion will become clearer. You will be able to plot the directional derivative as a function of the parameter *t*.

Technology Guidelines

NOTE: If you have just finished a module, restart *Mathematica* or close the *Kernel* before executing a new module.
TO OPEN CELLS, put your cursor on the right cell bracket and double click.
TO STOP AN EXECUTION
Select the *Kernel* pull-down menu and click on *Abort Evaluation.*
ORDER OF EXECUTION
Execute cells in the order given. Do not skip any Input cells within a given notebook.
SAVING NOTEBOOKS
You can save anytime to any directory you choose, and it is wise to save often.
However, before you do your final save, it is a good idea to delete all your output by selecting the *Delete All Output* selection under the *Kernel* pull-down menu.
EXPERIENCING MAJOR PROBLEMS
Save if appropriate, then shut down *Mathematica* and start it up again.

Part I: Skateboarding on a Flat Horizontal Surface

Imagine a skateboarder tracing out a figure-8 on a horizontal plane. The first set of commands that follow give the parametric equations of the skateboarder's path in the *xy*-plane. We'll start with the equations in polar form.

In[1]:=

```
Off[General::spell]

Off[General::spell1]

Clear[r, θ, t, x, y]

r[t_] := 16 Sin[t]^2

θ[t_] := t
```

Load the following special graphics package to do polar plots. Be sure to read it in BEFORE you try to execute **PolarPlot**.

In[6]:=

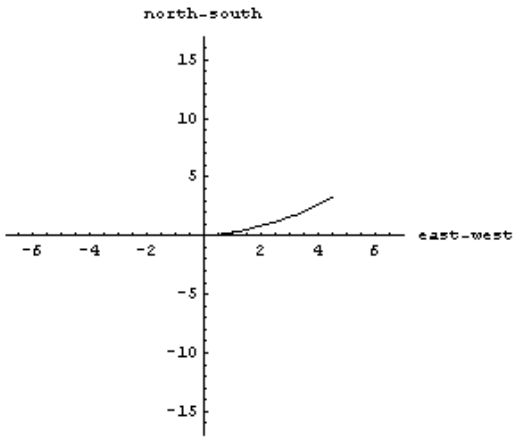
```
<< Graphics`Graphics`
```

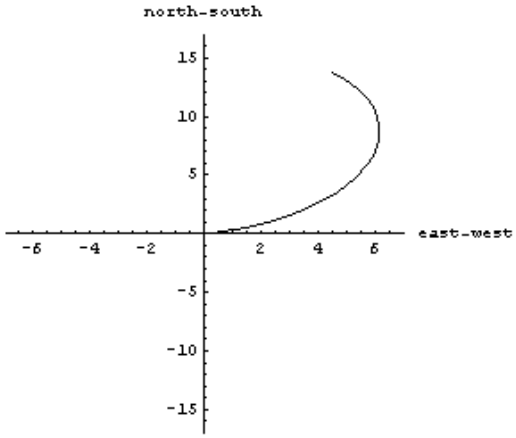
The following set of plots shows the progression of the skateboarder. To animate this set of plots, select any one of the plots in the sequence, then double-click on it. You can slow down the animation while it is playing by clicking on the appropriate button that appears in the lower left-hand corner. Notice how the skateboarder is tracing out the figure-8.

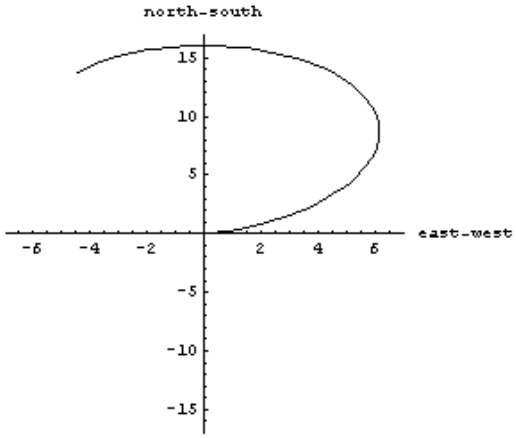
```
"> About Mathematica
```

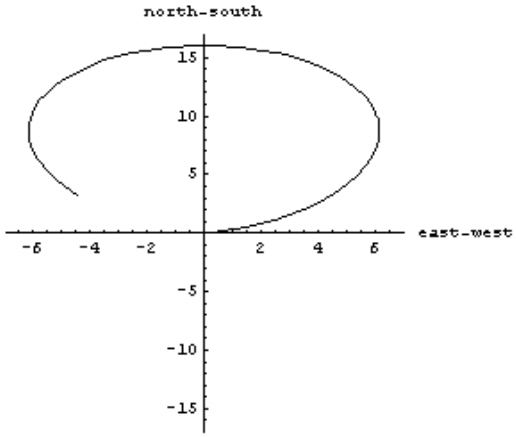
In[7]:=

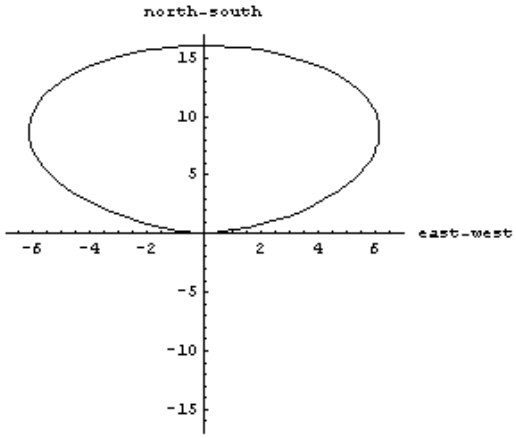
```
p[n_] := PolarPlot[r[t], {t, 0, 2π * n},
  PlotRange → {{-7, 7}, {-17, 17}}, AspectRati
  AxesLabel → {"east-west", "north-south"}]

Do[p[n], {n, 0.1, 1, 0.1}];
```

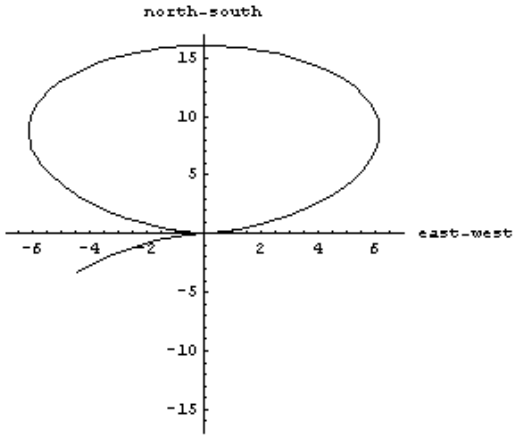


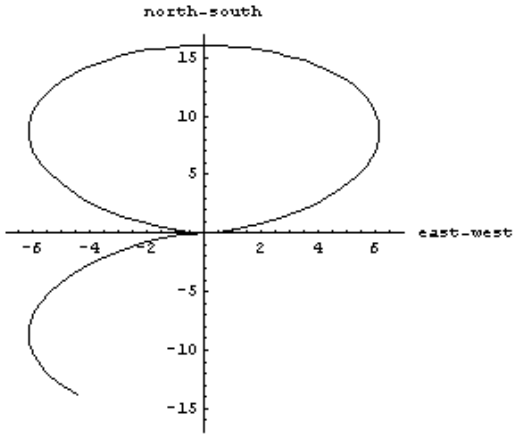

north-south
15
10
5
-6
-4
-2
2
4
6
east-west
-5
-10
-15

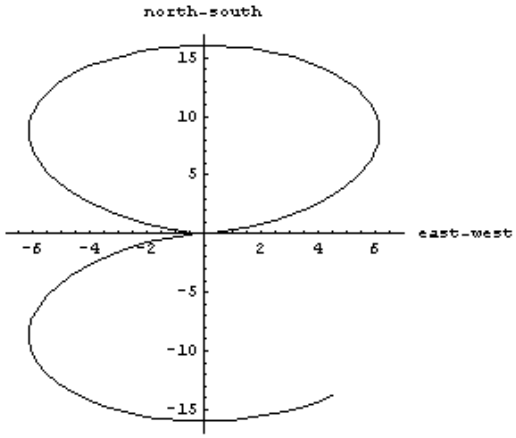

north-south
15
10
5
-6
-4
-2
2
4
6
east-west
-5
-10
-15

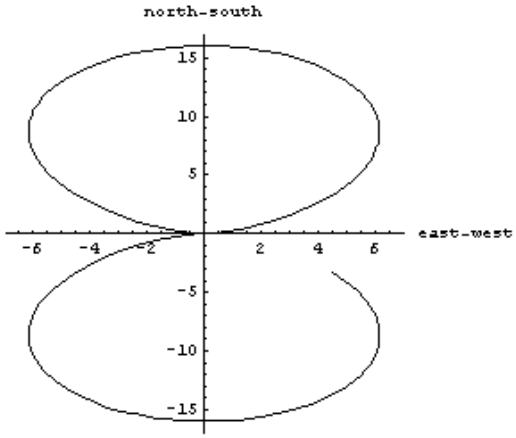

north-south
15
10
5
-6
-4
-2
2
4
6
east-west
-5
-10
-15

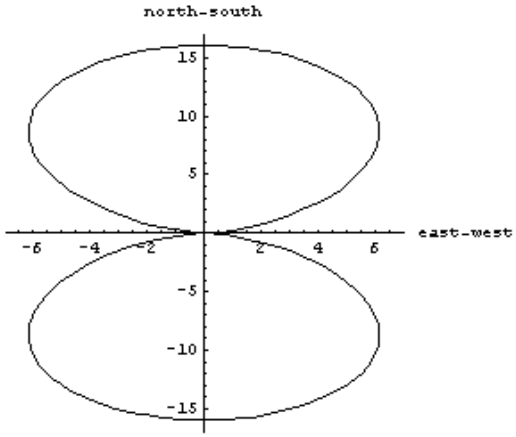

north-south
15
10
5
-6
-4
-2
2
4
6
east-west
-5
-10
-15


north-south
15
10
5
-6
-4
-2
2
4
6
east-west
-5
-10
-15


north-south
15
10
5
-6
-4
-2
2
4
6
east-west
-5
-10
-15







When we introduce vector quantities such as velocity and gradient, it is convenient to use rectangular

coordinates. For this reason we will redefine our motion and position, velocity, and unit tangent vectors parametrically in *x* and *y*, using the above parametrizations for r and θ.

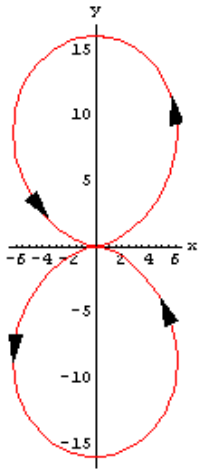
To show the direction of movement along the figure-8, we will place a few arrows on the graph. Adding arrows to the graph requires loading a graphics arrow package. Remember to load the package before you execute a command within the package.

In[9]:=

```
<< Graphics`Arrow`
```

In[10]:=

```
x1[t_] = r[t] Cos[θ[t]];

y1[t_] = r[t] Sin[θ[t]];

pp8 = ParametricPlot[{x1[t], y1[t]}, {t, 0, 2 π
  AspectRatio -> Automatic, PlotStyle -> RGBCo
  AxesLabel -> {"x", "y"},
  Epilog → {Arrow[{x1[1], y1[1]}, {x1[1.1], y1
     HeadScaling → Absolute],
    Arrow[{x1[2.5], y1[2.5]}, {x1[2.6], y1[2.
     HeadScaling → Absolute],
    Arrow[{x1[4], y1[4]}, {x1[4.1], y1[4.1]},
     HeadScaling → Absolute],
    Arrow[{x1[5.5], y1[5.5]}, {x1[5.6], y1[5.
     HeadScaling → Absolute]}];
```



In the following Parts as we examine the motion, we will need the unit tangent vector, so we now define the quantities we will use.

In[13]:=

```
position[t_] = {x1[t], y1[t]} // Simplify;

velocity[t_] = position'[t] // Simplify;

speed[t_] = √velocity[t].velocity[t] // Simp]

unittangent[t_] = velocity[t] / speed[t] // Sim

Print["position vector = ", position[t]]

Print["velocity vector = ", velocity[t]]

Print["unit tangent vector = ", unittangent[
```

```
position vector =
 {16 Cos[t] Sin[t]^2, 16 Sin[t]^3}

velocity vector =
 {-4 (Sin[t] - 3 Sin[3 t]), 48 Cos[t] Sin[t]^2}

unit tangent vector =
 {-(Sin[t] - 3 Sin[3 t]) / (2 √2 √((5 + 3 Cos[2 t]) Sin[t]^2)),
  3 √2 Cos[t] Sin[t]^2 / √((5 + 3 Cos[2 t]) Sin[t]^2)}
```

You Try It: Part I

Select and plot your own parametric equations for *x* and *y*. Replace the *x2* and *y2* functions in red. Remember that you are laying out a path for a skateboarder. You can execute the following example if you wish, and then try one of your own.

In[20]:=

```
x2[t_] = 10 Sin[t - 1]^2;
y2[t_] = 10 Cos[2 t]^3;
pos[t_] = {x2[t], y2[t]} // Simplify;
vel[t_] = pos'[t] // Simplify;
spd[t_] = √vel[t].vel[t] // Simplify;
unittan[t_] = vel[t] / spd[t] // Simplify;
Print["position vector = ", pos[t]]

Print["velocity vector = ", vel[t]]
```

```
Print["unit tangent vector = ", unittan[t]]
```

```
position vector =
 {10 Sin[1 - t]^2, 10 Cos[2 t]^3}
```

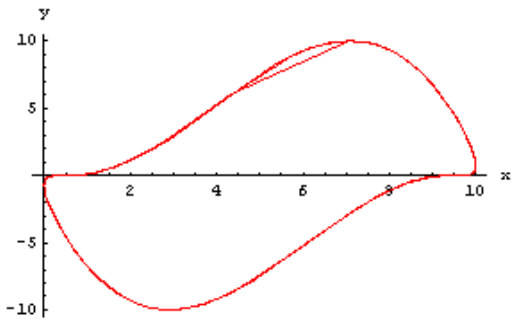
```
velocity vector =
 {-10 Sin[2 - 2 t], -60 Cos[2 t]^2 Sin[2 t]}
```

```
unit tangent vector =
 {-Sin[2 - 2 t] / Sqrt[Sin[2 - 2 t]^2 + 36 Cos[2 t]^4 Sin[2 t]^2],
  -6 Cos[2 t]^2 Sin[2 t] / Sqrt[Sin[2 - 2 t]^2 + 36 Cos[2 t]^4 Sin[2 t]^2]}
```

Look at the graph, and adjust the limits on *t* if necessary.

In[23]:=

```
ppnew = ParametricPlot[{x2[t], y2[t]}, {t, 0,
  PlotStyle -> RGBColor[1, 0, 0], AxesLabel ->
```



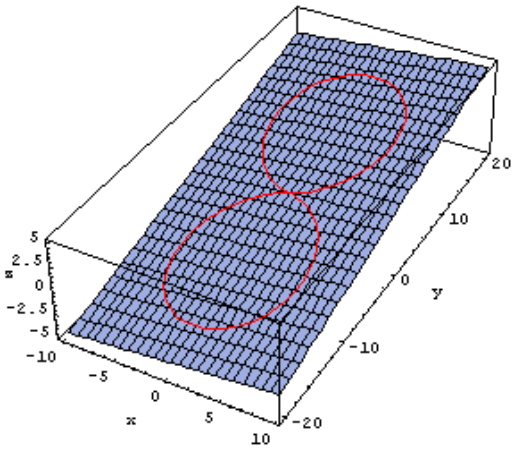
Part II: Skateboarding on a Planar Ramp and Computing and Analyzing the Directional Derivative

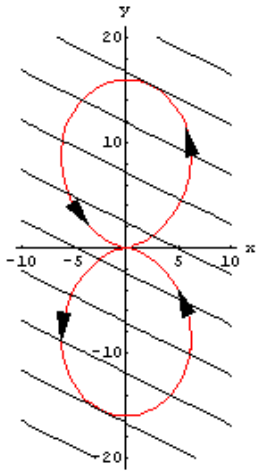
Now we consider a flat, inclined ramp for the skateboarder. The ramp has a 10% grade along the *x* direction (east-west) and a 20% grade along the *y* direction (north-south). Take the figure-8 from Part I as the horizontal projection of the skateboarder's path of motion on the inclined plane. If you were straight above the skateboarder and were looking down, the skateboarder would appear to move along the figure-8 in Part I. You would not see the incline of the plane. To visualize the ramp together with the figure-8, we produce plots, first in three-space, then projecting the motion onto the horizontal *xy*-plane.

In[24]:=

```
Clear[x, y]
```

```
ramp[x_, y_] = .1 x + .2 y;

p8ramp = ParametricPlot3D[
  {x1[t], y1[t], ramp[x1[t], y1[t]], RGBColor
  {t, 0, 2 π}, AxesLabel -> {x, y, z},
  DisplayFunction -> Identity];

pramp = Plot3D[ramp[x, y], {x, -10, 10}, {y, -2
  DisplayFunction -> Identity];

Show[p8ramp, pramp, DisplayFunction -> $Displ

ctramp = ContourPlot[ramp[x, y], {x, -10, 10},
  ContourShading -> False, DisplayFunction ->

Show[pp8, ctramp, DisplayFunction -> $Display
```





">About Mathematica

Look at the contour plot above. Recall that the gradient of a function of two variables is a vector in the domain perpendicular to the contours. If you dot the gradient of the ramp into a unit vector in the direction of movement around the figure-8 in the plane, can you predict when that dot product will be 0 and where it is positive or negative?

Let's calculate the directional derivative of the height of the ramp relative to the changing positions along the skateboarder's path in the domain. Note that we are interested only in the gradient of the z-function along the path of the figure-8. That is what the **/.{x->x1[t],y->y1[t]** refers to after the standard definition of the gradient.

Before computing the directional derivative, look at both the two- and three-dimensional plots. When might you expect the directional derivative to be 0? Why? When might you expect it to be a maximum or a minimum? Jot down your estimates and see how they compare to the computed results that follow.

In[31]:=

```
gradient[z_] := {D[z, x], D[z, y]} /. {x -> x1[t

dirderiv[z_, t_] := gradient[z].unittangent[t

ddramp[t_] = dirderiv[ramp[x, y], t]
```

Out[33]=

$$\frac{0.848528\,\text{Cos}[t]\,\text{Sin}[t]^2}{\sqrt{(5+3\,\text{Cos}[2\,t])\,\text{Sin}[t]^2}} - \frac{0.0353553\,(\text{Sin}[t]-3\,\text{Sin}[3\,t])}{\sqrt{(5+3\,\text{Cos}[2\,t])\,\text{Sin}[t]^2}}$$

Look at your result carefully. What happens to the value of the directional derivative when t is a multiple of π?

Let's look at both the left-hand (Direction→1) and the right-hand (Direction→-1) limits for the directional derivative as $t \to \pi$.

In[34]:=

```
Print["The left-hand limit is ",
 Limit[ddramp[t], t → π, Direction → 1]]

Print["The right-hand limit is ",
 Limit[ddramp[t], t → π, Direction → -1]]
```

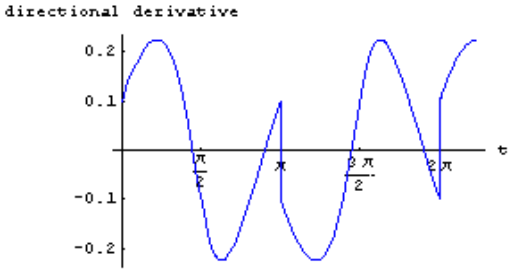
```
The left-hand limit is 0.1

The right-hand limit is -0.1
```

Uh-oh! What does this tell you about the value of the directional derivative at t = π?

Now consider the plot of this directional derivative as the skateboarder traverses the entire figure-8.

In[36]:=

```
pdd = Plot[ddramp[t], {t, 0, 7}, PlotStyle -> R
  AxesLabel -> {t, "directional derivative"},
  Ticks -> {{0, π/2, π, 3π/2, 2π, 5π/2}, Au
```



What do the sharp turns in this plot mean? Where are these sharp turns occurring? What has caused them? You may want to go back to your figure-8 graph and see what happens at $t = \pi$ and at $t = 2\pi$. The vertical segments drawn at $t = \pi$ and at $t = 2\pi$ don't really represent the directional derivative, because it actually takes a jump at those points. Is there any way that you could adjust the path of the skateboarder to make the directional derivative continuous and the motion smooth at $t = \pi$ and at $t = 2\pi$?

What does it mean when the directional derivative is 0? We will encounter this later in the module on Lagrange multipliers. For later comparison, we go ahead and use the **FindRoot** command to estimate the places where the directional derivative is 0, ignoring the places of discontinuity (t= π or 2π)

In[37]:=

```
dd1 = FindRoot[ddramp[t], {t, 1.5}];

dd2 = FindRoot[ddramp[t], {t, 2.8}];

dd3 = FindRoot[ddramp[t], {t, 4.5}];

dd4 = FindRoot[ddramp[t], {t, 6}];

Print[
 "The directional derivative is zero for the
   values of t: ", ddzero = {dd1, dd2, dd3, dd
```

```
The directional derivative is zero
   for the following values of t:
 {{t → 1.41379}, {t → 2.83495},
  {t → 4.55538}, {t → 5.97655}}
```

In[42]:=

```
Clear[xyzdisplay, xyzset]

xyzset =
 Table[N[{ddzero[[i, 1, 2]], x1[ddzero[[i, 1,
    y1[ddzero[[i, 1, 2]]],
    ramp[x1[ddzero[[i, 1, 2]]], y1[ddzero[[i,
  {i, 1, Length[ddzero]}];

xyzdisplay = PrependTo[xyzset, {"t", "x", "y",
```

Out[44]//TableForm=

t	x	y	z
1.4137869769742533`	2.4406706771430597`	15.416800904335451`	3.3274272485813965`
2.8349543944096314`	-1.3898630286206488`	0.44006509005973415`	-0.05097328485011807`
4.555379630564046`	-2.4406706771430717`	-15.416800904335446`	-3.3274272485813965`
5.976547047999425`	1.389863028620646`	-0.4400650900597327`	0.05097328485011804`

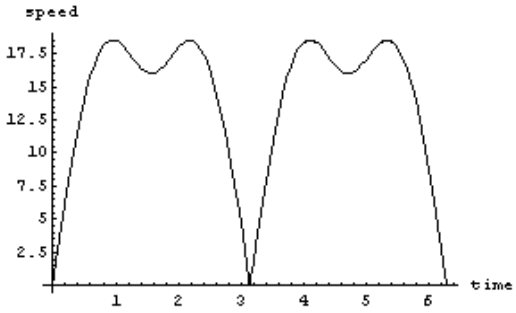
Where are these points on your plots?

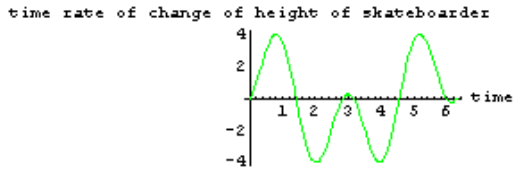
▪ Looking at the Time Derivative

If we assume that the parameter *t* represents time and determine the rate of change of the skateboarder's height with respect to time, we can multiply the directional derivative by the speed in the *xy*-plane. Why?

In[45]:=

```
timederivramp[t_] := ddramp[t] speed[t]
ps = Plot[speed[t], {t, 0, N[2 π]}, PlotRange -
   AxesLabel -> {time, speed}];
Plot[timederivramp[t], {t, 0, N[2 π]},
  AxesLabel ->
   {time, "time rate of change of height of
  PlotStyle -> RGBColor[0, 1, 0]];
```





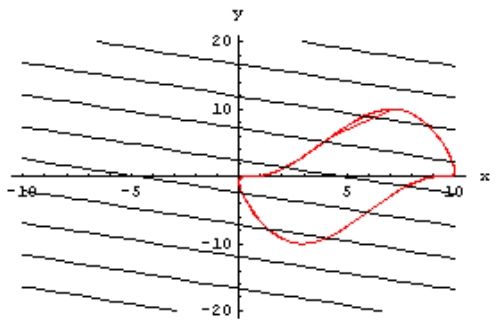
Interpret the speed plot. What is happening? Describe the skateboarder's motion. Why is the time derivative plot much smoother than the directional derivative plot?

You Try It: Part II

Take the curve that was defined in the You Try It: Part I, and picture it on this ramp.

In[48]:=

```
Show[ppnew, ctramp, DisplayFunction -> $Displ
```



Evaluate the directional derivative.

In[49]:=

```
gradient1[z_] := {D[z, x], D[z, y]} /. {x -> x2|
```

```
dirderiv1[z_, t_] := gradient1[z].unittan[t]

ddramp[t_] = dirderiv1[ramp[x, y], t]
```

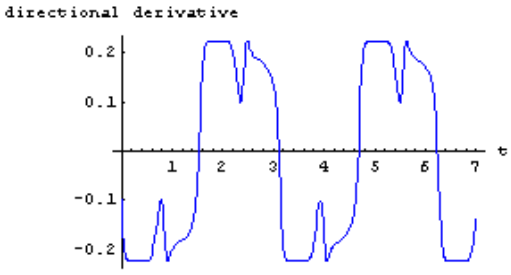
Out[51]=

$$-\frac{0.1\,\text{Sin}[2-2t]}{\sqrt{\text{Sin}[2-2t]^2+36\,\text{Cos}[2t]^4\,\text{Sin}[2t]^2}}-\frac{1.2\,\text{Cos}[2t]^2\,\text{Sin}[2t]}{\sqrt{\text{Sin}[2-2t]^2+36\,\text{Cos}[2t]^4\,\text{Sin}[2t]^2}}$$

Now compute the directional derivative.

In[52]:=

```
pdd = Plot[ddramp[t], {t, 0, 7}, PlotStyle -> R
  AxesLabel -> {t, "directional derivative"}]
```



What is happening when you get sharp points in your directional derivative plot? What does it mean when the directional derivative is 0?

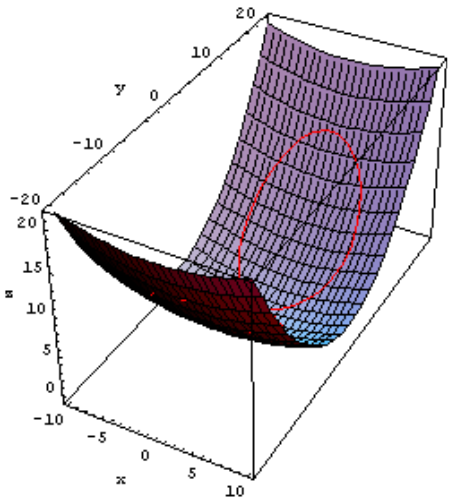
Part III: Skateboarding Inside an Elliptical Bowl

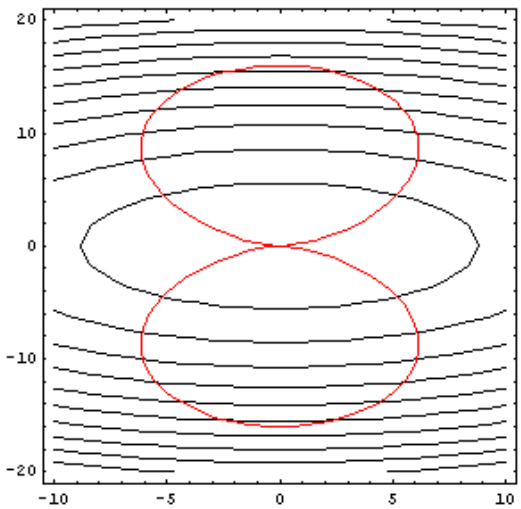
What if you were skateboarding on a bowl-shaped surface instead of on a planar ramp? Let's look at it graphically first.

In[53]:=

```
Clear[x, y]

bowl[x_, y_] := .02 x^2 + .05 y^2

p8bowl = ParametricPlot3D[
  {x1[t], y1[t], bowl[x1[t], y1[t]], RGBColor
  {t, 0, 2 π}, AxesLabel -> {x, y, z},
  DisplayFunction -> Identity];
```

```
pbowl = Plot3D[bowl[x, y] - .5, {x, -10, 10}, {
  AxesLabel -> {x, y, z}, DisplayFunction -> Id

Show[p8bowl, pbowl, DisplayFunction -> $Displ

ctbowl = ContourPlot[bowl[x, y], {x, -10, 10},
  ContourShading -> False, AxesLabel → {x, y},
  DisplayFunction -> Identity];

Show[ctbowl, pp8, DisplayFunction -> $Display
```





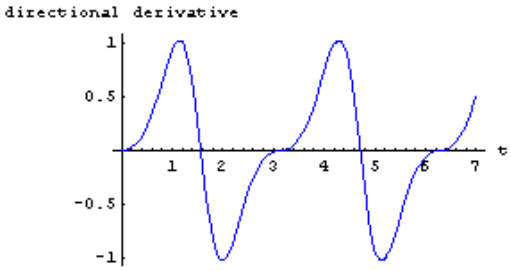
In case you are wondering why we subtracted .5 from the bowl height in the **Plot3D** command, it was done for display purposes only, so that the figure-8 plot would show up better on top of the surface. As you did for the ramp, estimate when the directional derivative will be 0 or will reach its maximum or minimum, based on the dot product interpretation. Then, examine your directional derivative.

In[60]:=

```
ddbowl[t_] = dirderiv[bowl[x, y], t]

Plot[ddbowl[t], {t, 0, 7}, PlotStyle -> RGBCol
 AxesLabel -> {t, "directional derivative"}];
```

Out[60]=

$$\frac{6.78823\,\text{Cos}[t]\,\text{Sin}[t]^5}{\sqrt{(5+3\,\text{Cos}[2\,t])\,\text{Sin}[t]^2}} - \frac{(0.226274\,\text{Cos}[t]\,\text{Sin}[t]^2\,(\text{Sin}[t]-3\,\text{Sin}[3\,t]))}{\left(\sqrt{(5+3\,\text{Cos}[2\,t])\,\text{Sin}[t]^2}\right)}$$



As before, consider what it means when the directional derivative is 0. Also at what places on the figure-8 will the skateboarder be the highest?

In[62]:=

```
dd1 = FindRoot[ddbowl[t], {t, 1.5}];

dd2 = FindRoot[ddbowl[t], {t, 2.8}];

dd3 = FindRoot[ddbowl[t], {t, 4.5}];

dd4 = FindRoot[ddbowl[t], {t, 6}];

Print[
 "The directional derivative is zero for the
   values of t: ", ddzero = {dd1, dd2, dd3, dd
```

```
The directional derivative is zero
   for the following values of t:
 {{t → 1.5708}, {t → 3.14159},
  {t → 4.71239}, {t → 6.28319}}
```

In[67]:=

```
xyzset =
 Table[N[{ddzero[[i, 1, 2]], x1[ddzero[[i, 1,
    y1[ddzero[[i, 1, 2]]],
    bowl[x1[ddzero[[i, 1, 2]]], y1[ddzero[[i,
  {i, 1, Length[ddzero]}];

xyzdisplay = PrependTo[xyzset, {"t", "x", "y",
```

Out[68]//TableForm=

t	x	y	z
1.5707963267948966`	9.796850830579018`*^-16	16.`	12.8`
3.1415926289534357`	-9.711201790470592`*^-15	2.3924863926897926`*^-22	1.8861488043047875`*^-30
4.71238898038469`	-2.9390552491737054`*^-15	-16.`	12.8`
6.283185268147552`	2.4375995773019434`*^-14	-9.514447137937293`*^-22	1.1883783398525271`*^-29

The directional derivative is 0 at each of the above points. Locate them on your surface and contour plots, and identify them as points of either maximum or minimum, or of neither.

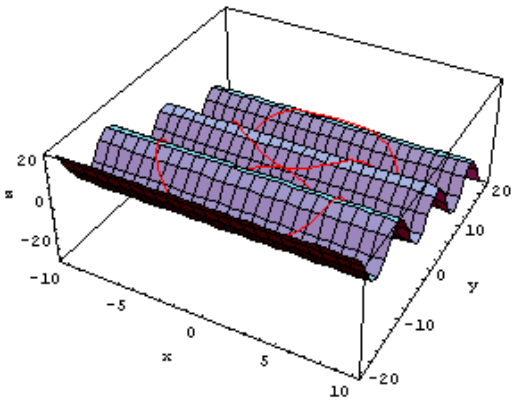
You Try It: Part III

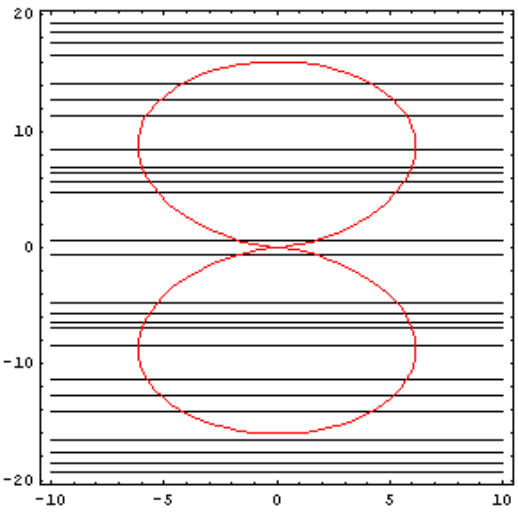
If you want to leave your path alone and just change your surface, alter the red command in the following roller coaster type surface. This roller coaster gives a much more complex directional derivative for you to analyze.

In[69]:=

```
r[t_] := 16 Sin[t]^2

θ[t_] := t

parx[t_] = r[t] Cos[θ[t]];
pary[t_] = r[t] Sin[θ[t]];
rc[x_, y_] := 10 Sin[y / 2] - y

p8rc = ParametricPlot3D[
  {parx[t], pary[t], rc[parx[t], pary[t]], R
  {t, 0, 2 π}, AxesLabel -> {x, y, z},
  DisplayFunction -> Identity];

prc = Plot3D[rc[x, y] - 1, {x, -10, 10}, {y, -20
  AxesLabel -> {x, y, z}, PlotRange -> All,
  DisplayFunction -> Identity];
```

```
Show[prc, p8rc, DisplayFunction -> $DisplayFu

ctrc = ContourPlot[rc[x, y], {x, -10, 10}, {y,
  ContourShading -> False, DisplayFunction ->

Show[ctrc, pp8, DisplayFunction -> $DisplayFu
```



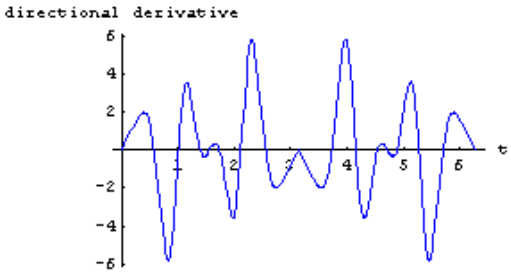


In[77]:=

```
ddrc[t_] = dirderiv[rc[x, y], t]

Plot[ddrc[t], {t, 0, N[2 π]}, PlotStyle -> RGB
 AxesLabel -> {t, "directional derivative"}];
```

Out[77]=

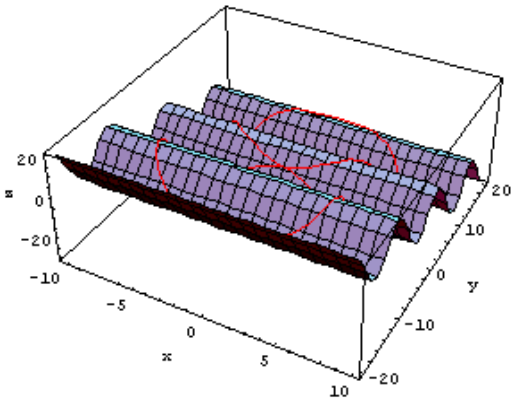
$$\frac{3\sqrt{2}\,\text{Cos}[t]\,(-1+5\,\text{Cos}[8\,\text{Sin}[t]^3])\,\text{Sin}[t]^2}{\sqrt{(5+3\,\text{Cos}[2\,t])\,\text{Sin}[t]^2}}$$

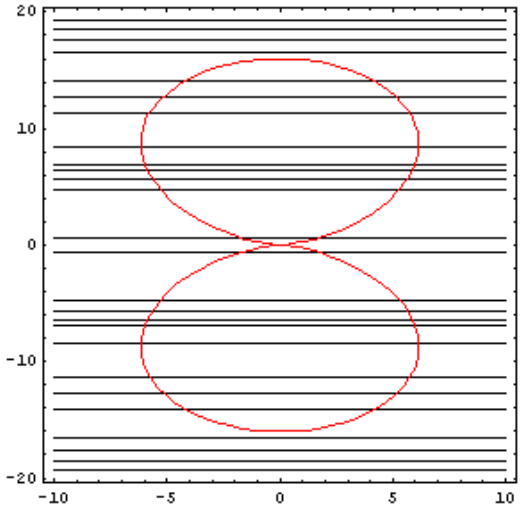


If you want to change both your path and your surface, just remember that the domain for the surface has to be adjusted to contain the domain for the path. Alter the commands in red.

In[79]:=

```
r[t_] := 16 Sin[t]^2

θ[t_] := t

parx[t_] = r[t] Cos[θ[t]];
pary[t_] = r[t] Sin[θ[t]];
rc[x_, y_] := 10 Sin[y / 2] - y

p8rc = ParametricPlot3D[
  {parx[t], pary[t], rc[parx[t], pary[t]], R
  {t, 0, 2 π}, AxesLabel -> {x, y, z},
  DisplayFunction -> Identity];

prc = Plot3D[rc[x, y] - 1, {x, -10, 10}, {y, -20
  AxesLabel -> {x, y, z}, PlotRange -> All,
  DisplayFunction -> Identity];

Show[prc, p8rc, DisplayFunction -> $DisplayFu

ctrc = ContourPlot[rc[x, y], {x, -10, 10}, {y,
  ContourShading -> False, DisplayFunction ->

Show[ctrc, pp8, DisplayFunction -> $DisplayFu
```





About *Mathematica*

Recall that the **PolarPlot** command expects as input *r* as a function of θ. In this case, since θ = t, we can simply use the r[t] defined above.
Go back.

In *Mathematica*, plots can be suppressed when first defined and then shown later with the **Show** command. This is done by calling upon the option "DisplayFunction→Identity" within the command that defines the plot and then calling upon the option "DisplayFunction→$DisplayFunction" when you are ready to **Show** the plot.
Go back.

Created by *Mathematica* (April 2, 2005)

W3C XHTML 1.1