

2D and 3D Measurements

©1999 Bill Davis, Horacio Porta and Jerry Uhl

Produced by Bruce Carpenter Published by Math Everywhere, Inc.

www.matheverywhere.com

VC.03 The Gradient Basics

B.1) The gradient

$$\text{gradf}[x, y] = \{D[f[x, y], x], D[f[x, y], y]\}$$

and the chain rule

$$D[f[x[t], y[t]], t] = \text{gradf}[x[t], y[t]] \cdot \{x'[t], y'[t]\}$$

Here is the chain rule for a function of one variable:

```
Clear[f, x, t]
D[f[x[t]], t]
f'[x[t]] x'[t]
```

This tells you that

$$\frac{df[x(t)]}{dt} = f'[x(t)] x'(t),$$

just as you have known for some time.

Here is the chain rule for a function of two variables:

```
Clear[f, x, t]
D[f[x[t], y[t]], t]
y'[t] f^{(0,1)}[x[t], y[t]] + x'[t] f^{(1,0)}[x[t], y[t]]
```

The notation

$$f^{(1,0)}[x, y] \text{ stands for } D[f[x, y], x]$$

and

$$f^{(0,1)}[x, y] \text{ stands for } D[f[x, y], y].$$

Some folks like to use the notation

$$\frac{\partial f[x, y]}{\partial x} = f^{(1,0)}[x, y] = D[f[x, y], x]$$

$$\frac{\partial f[x, y]}{\partial y} = f^{(0,1)}[x, y] = D[f[x, y], y]$$

So this output gives you three ways of writing the same thing:

$$\frac{df[x(t), y(t)]}{dt}$$

$$= D[f[x[t], y[t]], x] x'[t] + D[f[x[t], y[t]], y] y'[t]$$

$$= \frac{\partial f[x(t), y(t)]}{\partial x} x'[t] + \frac{\partial f[x(t), y(t)]}{\partial y} y'[t]$$

$$= f^{(1,0)}[x[t], y[t]] x'[t] + f^{(0,1)}[x[t], y[t]] y'[t].$$

This is the chain rule for functions of two variables.

□B.1.a.i) The gradient

How do you calculate the gradient of a function $f[x, y]$?

□Answer:

The gradient of $f[x, y]$ is a 2D vector given by:

```
Clear[f, gradf, x, y]
gradf[x_, y_] = {D[f[x, y], x], D[f[x, y], y]}
{f^{(1,0)}[x, y], f^{(0,1)}[x, y]}
```

In other words, the gradient of a function $f[x, y]$ is given by

$$\text{gradf}[x, y] = \{D[f[x, y], x], D[f[x, y], y]\}$$

$$= \left\{ \frac{\partial f[x, y]}{\partial x}, \frac{\partial f[x, y]}{\partial y} \right\}$$

$$= \{f^{(1,0)}[x, y], f^{(0,1)}[x, y]\}.$$

The gradient is important enough to carry the whole course from this point until its conclusion.

□B.1.a.ii)

How do you calculate the gradient of a function $f[x, y, z]$?

□Answer:

The gradient of $f[x, y, z]$ is a 3D vector given by:

```
Clear[f, gradf, x, y, z]
gradf[x_, y_, z_] = {D[f[x, y, z], x], D[f[x, y, z], y], D[f[x, y, z], z]}
{f^{(1,0,0)}[x, y, z], f^{(0,1,0)}[x, y, z], f^{(0,0,1)}[x, y, z]}
```

In other words, the gradient of a function $f[x, y, z]$ is given by

$$\text{gradf}[x, y, z] =$$

$$\{D[f[x, y, z], x], D[f[x, y, z], y], D[f[x, y, z], z]\}$$

$$= \left\{ \frac{\partial f[x, y, z]}{\partial x}, \frac{\partial f[x, y, z]}{\partial y}, \frac{\partial f[x, y, z]}{\partial z} \right\}$$

$$= \{f^{(1,0,0)}[x, y, z], f^{(0,1,0)}[x, y, z], f^{(0,0,1)}[x, y, z]\}.$$

Lots of times you'll see the notation

$$\nabla f[x, y] = \text{gradf}[x, y], \text{ or}$$

$$\nabla f[x, y, z] = \text{gradf}[x, y, z]$$

□B.1.b)

Here is Mathematica's calculation of the derivative of $f[x[t], y[t]]$ with respect to t :

```
Clear[f, x, y, t]
D[f[x[t], y[t]], t]
y'[t] f^{(0,1)}[x[t], y[t]] + x'[t] f^{(1,0)}[x[t], y[t]]
```

Most folks call this formula the "chain rule."

You can think of this as a dot product

$$\text{gradf}[x[t], y[t]] \cdot \{x'[t], y'[t]\}:$$

```
Clear[gradf]
gradf[x_, y_] = {D[f[x, y], x], D[f[x, y], y]}
gradf[x[t], y[t]] \cdot {x'[t], y'[t]}
y'[t] f^{(0,1)}[x[t], y[t]] + x'[t] f^{(1,0)}[x[t], y[t]]
```

Does the chain rule formula

$$D[f[x[t], y[t], z[t]], t]$$

$$= \text{gradf}[x[t], y[t], z[t]] \cdot \{x'[t], y'[t], z'[t]\}$$

work for functions of three variables?

□Answer:

Try it and see.

Here is $\text{gradf}[x, y, z]$:

```
Clear[f, x, y, z, t]
gradf[x_, y_, z_] = {D[f[x, y, z], x], D[f[x, y, z], y], D[f[x, y, z], z]}
{f^{(1,0,0)}[x, y, z], f^{(0,1,0)}[x, y, z], f^{(0,0,1)}[x, y, z]}
```

Here is $D[f[x[t], y[t], z[t]], t]$:

```
D[f[x[t], y[t], z[t]], t]
```

$$z'[t] f^{(0,0,1)}[x[t], y[t], z[t]] + y'[t] f^{(0,1,0)}[x[t], y[t], z[t]] + x'[t] f^{(1,0,0)}[x[t], y[t], z[t]]$$

Here is $\text{gradf}[x[t], y[t], z[t]] \cdot \{x'[t], y'[t], z'[t]\}$:

```
gradf[x[t], y[t], z[t]] \cdot {x'[t], y'[t], z'[t]}
z'[t] f^{(0,0,1)}[x[t], y[t], z[t]] + y'[t] f^{(0,1,0)}[x[t], y[t], z[t]] + x'[t] f^{(1,0,0)}[x[t], y[t], z[t]]
```

Check whether

$$D[f[x[t], y[t], z[t]], t] = \text{gradf}[x[t], y[t], z[t]] \cdot \{x'[t], y'[t], z'[t]\}$$

```
D[f[x[t], y[t], z[t]], t] ==
gradf[x[t], y[t], z[t]] \cdot {x'[t], y'[t], z'[t]}
True
```

You bet your sweet ear that the formula

$$D[f[x[t], y[t], z[t]], t] = \text{gradf}[x[t], y[t], z[t]] \cdot \{x'[t], y'[t], z'[t]\}$$

works for functions of three variables.

In fact, analogous versions of this formula work for functions of any number of variables.

□B.1.c.i)

Here is Mathematica's calculation of

$$D[f[\text{Sin}[3 t], e^{2 t}], t]$$

for a cleared function $f[x, y]$:

```
Clear[f, x, y, t]
x[t_] = Sin[3 t];
y[t_] = E^{2 t};
D[f[x[t], y[t]], t]
2 E^{2 t} f^{(0,1)}[Sin[3 t], E^{2 t}] + 3 Cos[3 t] f^{(1,0)}[Sin[3 t], E^{2 t}]
```

Use the fact that

$$D[f[x[t], y[t]], t] = \text{gradf}[x[t], y[t]] \cdot \{x'[t], y'[t]\}$$

to explain the Mathematica output.

□Answer:

```
Clear[gradf]
gradf[x_, y_] = {D[f[x, y], x], D[f[x, y], y]}
```

```
{f^(1,0)[x, y], f^(0,1)[x, y]}
Clear[tan]
tan[t] = D[{x[t], y[t]}, t]
{3 Cos[3 t], 2 E^2 t}
```

The chain rule says

$$D[f[x[t], y[t]], t] = \text{gradf}[x[t], y[t]] \cdot \text{tan}[t]$$

```
gradf[x[t], y[t]] \cdot tan[t]
2 E^2 t f^(0,1)[Sin[3 t], E^2 t] + 3 Cos[3 t] f^(1,0)[Sin[3 t], E^2 t]
```

Check:

```
D[f[x[t], y[t]], t]
2 E^2 t f^(0,1)[Sin[3 t], E^2 t] + 3 Cos[3 t] f^(1,0)[Sin[3 t], E^2 t]
```

Yes ma'am.

□B.1.c.ii)

Here is Mathematica's calculation of

$$D[f[x[t], y[t]], t]$$

for $f[x, y] = \sin[x^2 y^3]$ and cleared functions $x[t]$ and $y[t]$:

```
Clear[f, x, y, t]
f[x_, y_] = Sin[x^2 y^3];
D[f[x[t], y[t]], t]
Cos[x[t]^2 y[t]^3] (2 x[t] y[t]^3 x'[t] + 3 x[t]^2 y[t]^2 y'[t])
```

Use the fact that

$$D[f[x[t], y[t]], t] = \text{gradf}[x[t], y[t]] \cdot \{x'[t], y'[t]\}$$

to explain the Mathematica output.

□ Answer:

```
Clear[gradf]
gradf[x_, y_] = {D[f[x, y], x], D[f[x, y], y]}
{2 x y^3 Cos[x^2 y^3], 3 x^2 y^2 Cos[x^2 y^3]}
Clear[tan]
tan[t] = D[{x[t], y[t]}, t]
{x'[t], y'[t]}
```

The chain rule says

$$D[f[x[t], y[t]], t] = \text{gradf}[x[t], y[t]] \cdot \text{tan}[t]$$

```
gradf[x[t], y[t]] \cdot tan[t]
```

```
2 Cos[x[t]^2 y[t]^3] x[t] y[t]^3 x'[t] + 3 Cos[x[t]^2 y[t]^3] x[t]^2 y[t]^2 y'[t]
```

Check:

```
Expand[D[f[x[t], y[t]], t]]
2 Cos[x[t]^2 y[t]^3] x[t] y[t]^3 x'[t] + 3 Cos[x[t]^2 y[t]^3] x[t]^2 y[t]^2 y'[t]
```

You can do these by hand with no sweat.

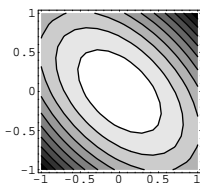
B.2) Level curves, level surfaces and the gradient as normal vector

Anytime you are examining a function, you can learn a lot by plotting some of its level curves.

```
Clear[f, x, y]
f[x_, y_] = 5 - (x^2 + x y + y^2)
5 - x^2 - x y - y^2
```

Here is a plot of some level curves $f[x, y] = c$ for various c 's as selected by Mathematica. You are looking down at the surface $z = f[x, y]$ from the positive z -direction.

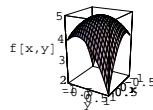
```
a = -1;
b = 1;
levelcurves = ContourPlot[Evaluate[f[x, y]], {x, a, b}, {y, a, b},
ContourSmoothing -> Automatic, AxesLabel -> {"x", "y"}];
```



The lighter shading indicates larger values of $f[x, y]$.

Here is the 3D plot of the surface $z = f[x, y]$:

```
surfaceplot = ParametricPlot3D[{x, y, f[x, y]}, {x, a, b},
{y, a, b}, ViewPoint -> CMView, AxesLabel -> {"x", "y", "f[x,y]"}];
```

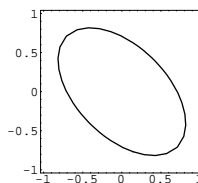


The level curves are plots of the shapes of various trips on the surface that keep the height $f[x, y]$ constant for the whole trip.

Here is a plot of the shape of the trip that keeps

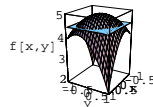
$$f[x, y] = 4.5:$$

```
c = 4.5;
Show[levelcurves, Contours -> {c}, ContourShading -> False];
```



Here is the actual plot of the same trip which lies on the intersection of the surface and the plane $z = 4.5$:

```
c = 4.5;
plane = ParametricPlot3D[{x, y, c}, {x, a, b},
{y, a, b}, PlotPoints -> {2, 2}, DisplayFunction -> Identity];
Show[surfaceplot, plane, ViewPoint -> CMView,
DisplayFunction -> $DisplayFunction];
```



□B.2.a) The gradients are perpendicular to level curves

Here's a new function

$$f[x, y] = 2x^2 + xy + y^2$$

together with its gradient:

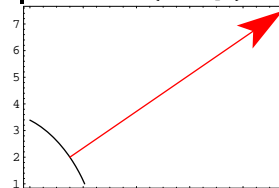
```
Clear[x, y, f, gradf]
f[x_, y_] = 2 x^2 + x y + y^2;
gradf[x_, y_] = {D[f[x, y], x], D[f[x, y], y]}
{4 x + y, x + 2 y}
```

Here is a part of the level curve

$$f[x, y] = f[1.5, 2]$$

shown with $\text{gradf}[1.5, 2]$ plotted with its tail at $\{1.5, 2\}$:

```
levelcurve =
ContourPlot[f[x, y], {x, 0, 3}, {y, 1, 4}, Contours -> {f[1.5, 2]},
ContourShading -> False, DisplayFunction -> Identity];
point = {1.5, 2};
gradient = Arrow[gradf[1.5, 2], Tail -> point, VectorColor -> Red];
Show[levelcurve, gradient, PlotRange -> All, AspectRatio -> Automatic,
AxesLabel -> {"x", "y"}, DisplayFunction -> $DisplayFunction];
```



That gradient vector is perpendicular to the level curve. Is this just an accident?

□ Answer:

This is no accident.

In Mathematics, there are no accidents.

Here's why:

If you parameterize the level curve

$$f[x, y] = c$$

with a parameterization {x[t], y[t]} and then you put

$$g[t] = f[x[t], y[t]],$$

then you get g[t] = c no matter what t is.

Consequently

$$g'[t] = 0$$

no matter what t is. But the chain rule says

$$g'[t] = \text{grad}f[x[t], y[t]] \cdot \{x'[t], y'[t]\}.$$

So

$$\text{grad}f[x[t], y[t]] \cdot \{x'[t], y'[t]\} = 0$$

no matter what t is. This means that the gradient

$$\text{grad}f[x[t], y[t]]$$

is perpendicular to the tangent vector

$$\{x'[t], y'[t]\}$$

at {x[t], y[t]}.

As a result, no matter what {x, y} you go to on a level curve

$$f[x, y] = c,$$

the gradient gradf[x, y] is perpendicular to the level curve at {x, y}.

Read that again!

Check it out again, going with

$$f[x, y] = x^2 + 4y^2:$$

Here is a true-scale plot of the level curve

$$f[x, y] = x^2 + 4y^2 = 36$$

and some of the gradients gradf[x, y] with tails at some points {x, y} on the level curve.

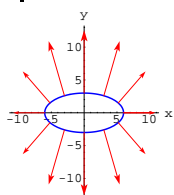
The level curve

$$f[x, y] = x^2 + 4y^2 = 36$$

is the ellipse

$$\left(\frac{x}{6}\right)^2 + \left(\frac{y}{3}\right)^2 = 1.$$

```
Clear[x, y, f, gradf]
f[x_, y_] = x^2 + 4 y^2;
gradf[x_, y_] = {D[f[x, y], x], D[f[x, y], y]};
Clear[t]
{x[t_], y[t_]} = {6 Cos[t], 3 Sin[t]};
levelcurve = ParametricPlot[{x[t], y[t]}, {t, 0, 2 Pi},
  PlotStyle -> {Blue, Thickness[0.01]}, DisplayFunction -> Identity];
scalefactor = 0.4;
gradients = Table[Arrow[gradf[x[t], y[t]], Tail -> {x[t], y[t]},
  VectorColor -> Red, ScaleFactor -> scalefactor], {t, 0, 2 Pi, 2 Pi/12}];
Show[levelcurve, gradients,
  AspectRatio -> Automatic, AxesLabel -> {"x", "y"}, PlotRange -> All,
  DisplayFunction -> $DisplayFunction];
```



Perpendicular as all get-out.

It will work any time and place.

□B.2.b)

Does this work in three dimensions?

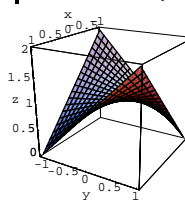
□Answer:

Check it out.

Here is part of the level surface

$$f[x, y, z] = z - x y = 1.$$

```
Clear[f, gradf, x, y, z]
f[x_, y_, z_] = z - x y;
gradf[x_, y_, z_] =
  {D[f[x, y, z], x], D[f[x, y, z], y], D[f[x, y, z], z]};
Solve[f[x, y, z] == 1, z]
{{z -> 1 + x y}}
levelsurface = ParametricPlot3D[{x, y, 1 + x y}, {x, -1, 1},
  {y, -1, 1}, ViewPoint -> CMView, BoxRatios -> Automatic,
  AxesLabel -> {"x", "y", "z"}];
```

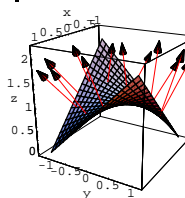


You can get one of the many curves on this surface by taking the surface plotting function {x, y, 1 + x y} and setting

$$x = \frac{\text{Cos}[t]}{2} \text{ and } y = \text{Sin}[t].$$

Here come a lot of gradients with tails on this curve:

```
Clear[x, y, z, t]
{x[t_], y[t_], z[t_]} = {x, y, 1 + x y} /. {x -> Cos[t]/2, y -> Sin[t]};
gradients = Table[Arrow[gradf[x[t], y[t], z[t]],
  Tail -> {x[t], y[t], z[t]}, VectorColor -> Red], {t, 0, 2 Pi, 2 Pi/12}];
Show[levelsurface, gradients];
```



The gradients are all perpendicular to the surface—just as normal as Beaver, Wally, June, and Ward.

B.3) The gradient points in the direction of greatest initial increase

The negative gradient points in the direction of greatest initial decrease

□B.3.a)

Here is a function and its gradient:

```
Clear[x, y, f, gradf]
f[x_, y_] = -5 x^2 + (x^4 y)/4 + 2 y + 10;
gradf[x_, y_] = {D[f[x, y], x], D[f[x, y], y]};
{-10 x + x^3 y, 2 + (x^4)/4}
```

You are sitting at:

```
{a, b} = {1.5, 1.8}
{1.5, 1.8}
```

At this point the function's value is:

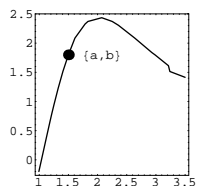
```
f@@{a, b}
4.62813
```

The instruction Apply[f, {a, b}] accomplishes the same thing as the instruction f[a, b]:

```
f[a, b]
4.62813
```

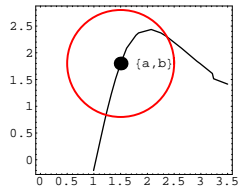
Here is a plot of part of the level curve f[x, y] = f[a, b]:

```
levelcurve = ContourPlot[f[x, y], {x, a - 2, a + 2},
  {y, b - 2, b + 2}, Contours -> {f[a, b]}, ContourShading -> False,
  ContourSmoothing -> True, DisplayFunction -> Identity];
labels = {Graphics[{PointSize[0.07], Point[{a, b}]}],
  Graphics[Text["{a,b}", {a, b}, {-1.5, 0}]}];
Show[levelcurve, labels, AspectRatio -> Automatic,
  DisplayFunction -> $DisplayFunction];
```



You can leave the {a, b} in the direction of any point on the circle of radius 1 centered at {a, b}.

```
circle = Graphics[{Red, Thickness[0.01], Circle[{a, b}, 1]}];
Show[levelcurve, labels, circle, AspectRatio -> Automatic,
  DisplayFunction -> $DisplayFunction];
```



Which direction should you go to get the greatest initial increase of the function?
 Which direction should you go to get the greatest initial decrease of the function?

□Answer:

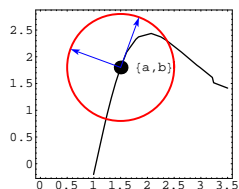
Calculus&Mathematica thanks 1991 C&M student Tony Pulokas of the University of Illinois for suggesting this way of looking at the problem.

Two unit vectors come to mind. They are unit tangent and unit normal vectors to the level curve at {a, b}. One unit normal is the unit vector in the direction of the gradient.

```
unitgradnormal =  $\frac{\text{gradf}[a, b]}{\sqrt{\text{gradf}[a, b] \cdot \text{gradf}[a, b]}}$ 
{-0.93911, 0.343617}
```

You can get a unit tangent vector by switching the components and hitting one of them with a minus one.

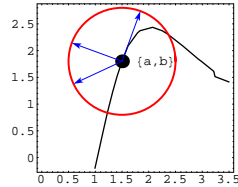
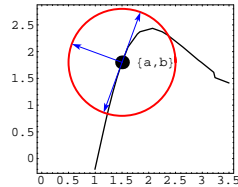
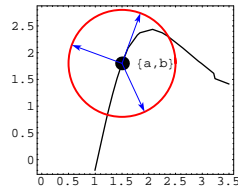
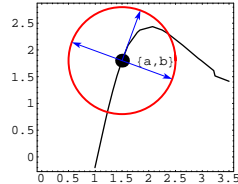
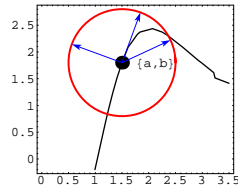
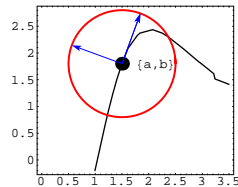
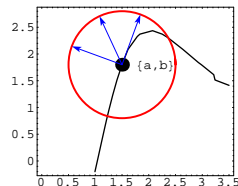
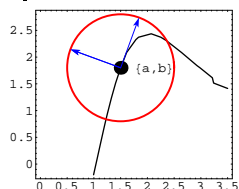
```
unittan = {unitgradnormal[[2]], -unitgradnormal[[1]]}
{0.343617, 0.93911}
setup = Show[levelcurve, labels, circle,
  Arrow[unitgradnormal, Tail -> {a, b}],
  Arrow[unittan, Tail -> {a, b}],
  AspectRatio -> Automatic, DisplayFunction -> $DisplayFunction];
```



You can leave the point in the direction of any vector with tail at {a, b} and tip on the circle of radius 1 centered at {a, b}.

Here are some possibilities:

```
Table[Show[setup,
  Arrow[Cos[t] unitgradnormal + Sin[t] unittan, Tail -> {a, b}],
  DisplayFunction -> $DisplayFunction, AspectRatio -> Automatic],
  {t, 0, 2 Pi -  $\frac{\pi}{4}$ ,  $\frac{\pi}{4}$ };
```



Animate these and run at a brisk speed.

In the movie, you saw the tips of the vectors

$\text{Cos}[t] \text{unitnormal} + \text{Sin}[t] \text{unittan}$
 with tails at {a, b} sweep out the circle as t went from 0 to 2π . The question is how to set t so that the vector

$\text{Cos}[t] \text{unitnormal} + \text{Sin}[t] \text{unittan}$
 with tail at {a, b} points in the direction of greatest initial change of $f[x, y]$ as you leave {a, b}.

Think for a second and you will probably agree that going in the

direction of unitnormal is a bad idea because it snuggles against the level curve and $f[x, y]$ doesn't change at all on this level curve.

In fact any direction that has a nonzero tangential component is a poor choice because the tangential component won't do much for changing the function.

To get the greatest initial change in $f[x, y]$ you should leave $\{a, b\}$ in the direction of

$$\text{Cos}[t] \text{unitnormal} + \text{Sin}[t] \text{unittan}$$

with t set so that the unittan term is zeroed out. This means you set

$t = 0$ or π and this leaves you the choice of directions:

$$\text{unitnormal}$$

or in the direction of

$$-\text{unitnormal}.$$

Because the unitnormal points in the same direction as the gradient, to

get the greatest initial change in $f[x, y]$ you should leave $\{a, b\}$ in the direction of either

$$\text{gradf}[a, b] \text{ or } -\text{gradf}[a, b].$$

Let's see which one:

```
Clear[s]
{f@@({a, b} + s gradf[a, b]),
 f@@({a, b}, f@@({a, b} + s (-gradf[a, b])))} /.
 s -> 0.1
{12.4803, 4.62813, -3.60418}
{f@@({a, b} + s gradf[a, b]),
 f@@({a, b}, f@@({a, b} + s (-gradf[a, b])))} /.
 s -> 0.01
{5.529, 4.62813, 3.72469}
{f@@({a, b} + s gradf[a, b]),
 f@@({a, b}, f@@({a, b} + s (-gradf[a, b])))} /.
 s -> 0.001
```

```
{4.71843, 4.62813, 4.53779}
{f@@({a, b} + s gradf[a, b]),
 f@@({a, b}, f@@({a, b} + s (-gradf[a, b])))} /.
 s -> 0.0001
{4.63716, 4.62813, 4.61909}
```

The gradient points in the direction of greatest initial increase.

The negative gradient points in the direction of the greatest initial

decrease.

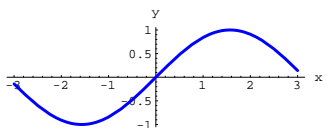
This will happen for any function at any point at which the gradient is not $\{0, 0\}$.

B.4) Using linearizations to help to explain the chain rule

$$D[f[x[t], y[t]], t] = \text{gradf}[x[t], y[t]] \cdot \{x'[t], y'[t]\}$$

Here is a simple curve in two dimensions:

```
Clear[f, x]
f[x_] = Sin[x];
fplot = Plot[f[x], {x, -3, 3}, PlotStyle -> {{Blue, Thickness[0.01]}},
 AxesLabel -> {"x", "y"}];
```



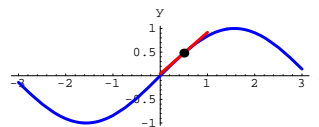
Here's the linearization of $f[x]$ at $\{a, f[a]\}$:

```
Clear[linearf, a]
linearf[x_, a_] = f[a] + f'[a] (x - a)
(-a + x) Cos[a] + Sin[a]
```

And a plot in the case $a = 0.5$:

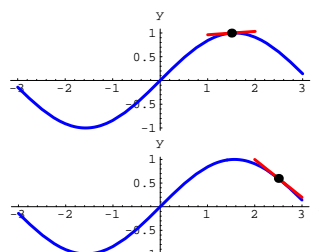
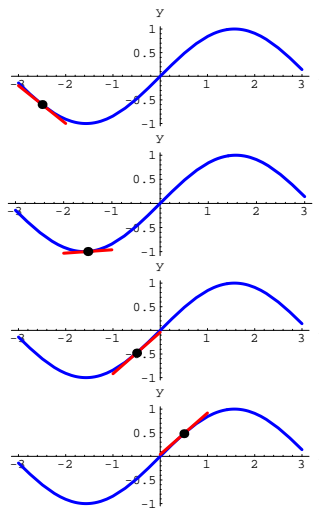
```
Clear[a, linplot]
linplot[a_] := Plot[linearf[x, a], {x, a - 0.5, a + 0.5},
 PlotStyle -> {{Red, Thickness[0.01]}}, DisplayFunction -> Identity];
pointplot[a_] = Graphics[{PointSize[0.03], Point[{a, f[a]}]}];

Show[fplot, linplot[0.5], pointplot[0.5],
 DisplayFunction -> $DisplayFunction];
```



Here's a little movie:

```
Table[Show[fplot, linplot[a],
 pointplot[a], DisplayFunction -> $DisplayFunction],
 {a, -2.5, 2.5}];
```



That's right; $\text{linearf}[x, a]$ is nothing but the tangent line at $x = a$.

To linearize a function $f[x, y]$, you can use the gradient.

The gradient of a function $f[x, y]$ is given by:

```
Clear[f, x, y, gradf]
gradf[x_, y_] = {D[f[x, y], x], D[f[x, y], y]}
{f^(1,0)[x, y], f^(0,1)[x, y]}
```

Just as the linearization of a function $f[x]$ at a point $x = a$ is given by

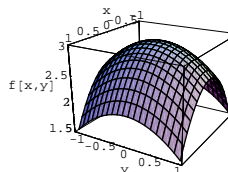
$$\text{linearf}[x, a] = f[a] + f'[a](x - a),$$

the linearization of a function $f[x, y]$ at a 2D point $\{a, b\}$ is given by

$$\text{linearf}[x, y, \{a, b\}] = f[a, b] + \text{gradf}[a, b] \cdot \{x - a, y - b\}.$$

To help you stick your mental tongs into the idea behind using the gradient for linearization, look at the following plot of a function $f[x, y]$ and a point $\{a, b\}$:

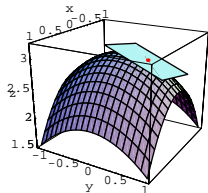
```
Clear[f, x, y]
f[x_, y_] = 3 - x^2 - 0.5 y^2;
actualsurfaceplot =
 ParametricPlot3D[{x, y, f[x, y]}, {x, -1, 1}, {y, -1, 1},
 Axes -> Automatic, AxesLabel -> {"x", "y", "f[x,y]"}, PlotRange -> All,
 ViewPoint -> CMView, DisplayFunction -> $DisplayFunction];
```



Now throw in the plot of
`linearf[x, y, {a, b}]`
 at the indicated point $\{a, b, f[a, b]\}$:

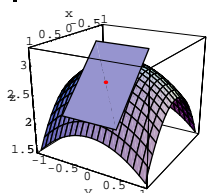
```
Clear[a, b, gradf, linearf, point];
gradf[x_, y_] := {D[f[x, y], x], D[f[x, y], y]};
linearf[x_, y_, {a_, b_}] := f[a, b] + gradf[a, b] . {x - a, y - b};
linearizedplot[a_, b_] :=
  ParametricPlot3D[{x, y, linearf[x, y, {a, b}]}, {x, a - 0.5, a + 0.5},
    {y, b - 0.5, b + 0.5}, PlotPoints -> {2, 2}, DisplayFunction -> Identity];
point[a_, b_] :=
  Graphics3D[{Red, PointSize[0.02], Point[{a, b, f[a, b]}]}];

{a, b} = {-0.1, 0.4};
Show[actualsurfaceplot, linearizedplot[a, b], point[a, b],
  Axes -> Automatic, AxesLabel -> {"x", "y", "z"}, PlotRange -> All,
  ViewPoint -> CMView, DisplayFunction -> $DisplayFunction];
```

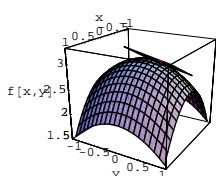


The plot of `linearf[x, y, {a, b}]` is that little bit of plane tangent to the surface at the plotted point $\{a, b, f[a, b]\}$.
 See what happens for other some other points:

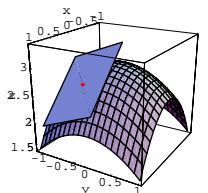
```
{a, b} = {0.5, 0};
Show[actualsurfaceplot, linearizedplot[a, b], point[a, b],
  Axes -> Automatic, AxesLabel -> {"x", "y", "z"}, PlotRange -> All,
  ViewPoint -> CMView, DisplayFunction -> $DisplayFunction];
```



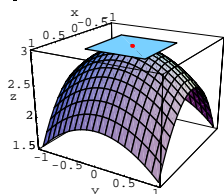
```
{a, b} = {-0.25, 0.2};
Show[actualsurfaceplot, linearizedplot[a, b], point[a, b],
  Axes -> Automatic, AxesLabel -> {"x", "y", "f[x,y]"}, PlotRange -> All,
  ViewPoint -> CMView, DisplayFunction -> $DisplayFunction];
```



```
{a, b} = {0.5, -0.4};
Show[actualsurfaceplot, linearizedplot[a, b], point[a, b],
  Axes -> Automatic, AxesLabel -> {"x", "y", "z"}, PlotRange -> All,
  ViewPoint -> CMView, DisplayFunction -> $DisplayFunction];
```



```
{a, b} = {0, 0};
Show[actualsurfaceplot, linearizedplot[a, b], point[a, b],
  Axes -> Automatic, AxesLabel -> {"x", "y", "z"}, PlotRange -> All,
  ViewPoint -> CMView, DisplayFunction -> $DisplayFunction];
```



Play with some other choices of a and b until you are satisfied.

□B.4.a) Why the linearized plot is tangent to the actual plot

In all the experiments, the plot of `linearf[x, y, {a, b}]` is tangent to the surface at $\{a, b, f[a, b]\}$.

Explain why this will always happen, no matter what function $f[x, y]$ and point $\{a, b, f[a, b]\}$ you go with.

□Answer:

One possible explanation is that at the point $\{a, b, f[a, b]\}$ the surface

$$z = f[x, y]$$

and the plane

$$z = \text{linearf}[x, y, \{a, b\}] = f[a, b] + \text{gradf}[a, b] \cdot \{x - a, y - b\}.$$

have parallel normal vectors.

To investigate this, run with cleared functions to see what these normal vectors are:

```
Clear[f, x, y, a, b, gradf, linearf, point];
gradf[x_, y_] := {D[f[x, y], x], D[f[x, y], y]};
linearf[x_, y_, {a_, b_}] := f[a, b] + gradf[a, b] . {x - a, y - b};
f[a, b] + (-b + y) f^{(0,1)}[a, b] + (-a + x) f^{(1,0)}[a, b]
```

An xyz-equation of the plot of

$$z = \text{linearf}[x, y, \{a, b\}]$$

is

$$f^{(1,0)}[a, b](x - a) + f^{(0,1)}[a, b](y - b) - (z - f[a, b]) = 0.$$

This is the equation of a plane with normal vector

$$\{f^{(1,0)}[a, b], f^{(0,1)}[a, b], -1\}.$$

Now check out the normal to the actual surface at a point $\{a, b, f[a, b]\}$:

If this mystifies you, look at the Basics in the last lesson.

```
Clear[surfacepoint, xcurve, ycurve, normal];
surfacepoint[x_, y_] := {x, y, f[x, y]};
xcurve[x_] := surfacepoint[x, b];
ycurve[y_] := surfacepoint[a, y];
tanxcurve[x_] := D[xcurve[x], x];
tanycurve[y_] := D[ycurve[y], y];
normal = tanxcurve[a] * tanycurve[b];
{-f^{(1,0)}[a, b], -f^{(0,1)}[a, y], 1}
```

This is just the negative of the normal vector

$$\{f^{(1,0)}[a, b], f^{(0,1)}[a, b], -1\}$$

to the plot of `linearf[x, y, {a, b}]` found above.

The upshot:

The plot of `linearf[x, y, {a, b}]` has no choice but to be tangent to the surface at the point $\{a, b, f[a, b]\}$.

□B.4.b.i)

You are given a curve $\{x[t], y[t]\}$ in two dimensions and a function $f[x, y]$. You can put

$$z[t] = f[x[t], y[t]]$$

You can also linearize $f[x, y]$ at a point

$$\{x[t_0], y[t_0], f[x[t_0], y[t_0]]\}$$

and look at the new function

$$z\text{new}[t] = \text{linearf}[x[t], y[t], \{x[t_0], y[t_0]\}].$$

Here's what happens in the specific case with

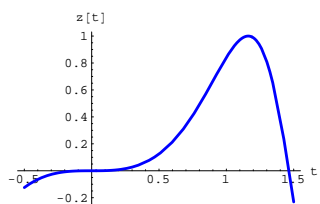
$$f[x, y] = \text{Sin}[x y]$$

and

$$\{x[t], y[t]\} = \{t, t^2\}$$

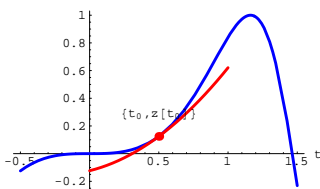
for some sample selections of t_0 :

```
Clear[f, x, y, z, t];
f[x_, y_] := Sin[x y];
{x[t_], y[t_]} = {t, t^2};
z[t_] := f[x[t], y[t]];
actualcurve = Plot[z[t], {t, -0.5, 1.5},
  PlotStyle -> {{Blue, Thickness[0.01]}}, AxesLabel -> {"t", "z[t]"}];
```



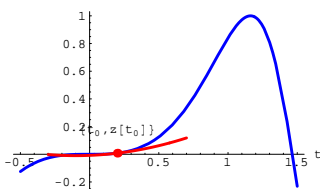
Here's what happens when you take $t_0 = 0.5$ and plot the new curve $z_{new}[t] = \text{linearf}[x[t], y[t], \{x[t_0], y[t_0]\}]$ and show the two plots together:

```
Clear[a, b, gradf, linearf, point, znew];
gradf[x_, y_] = {D[f[x, y], x], D[f[x, y], y]};
linearf[x_, y_, {a_, b_}] := f[a, b] + gradf[a, b] . {x - a, y - b};
znew[t_] = linearf[x[t], y[t], {a, b}];
point[t_] =
  {Graphics[Text["{"t_0", z[t_0]}", {t, z[t]}, {0, -3}],
   Graphics[{Red, PointSize[0.03], Point[{t, znew[t]}]}]};
t0 = 0.5;
{a, b} = {x[t0], y[t0]};
newcurve = Plot[znew[t], {t, t0 - 0.5, t0 + 0.5},
  PlotStyle -> {{Red, Thickness[0.01]}}, DisplayFunction -> Identity];
Show[actualcurve, newcurve, point[t0], PlotRange -> All,
  AxesLabel -> {"t", ""}, DisplayFunction -> $DisplayFunction];
```



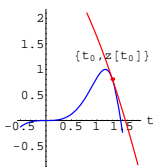
Try it again for different selections of t_0 :

```
t0 = 0.2;
{a, b} = {x[t0], y[t0]};
newcurve = Plot[znew[t], {t, t0 - 0.5, t0 + 0.5},
  PlotStyle -> {{Red, Thickness[0.01]}}, DisplayFunction -> Identity];
Show[actualcurve, newcurve, point[t0], PlotRange -> All,
  AxesLabel -> {"t", ""}, DisplayFunction -> $DisplayFunction];
```



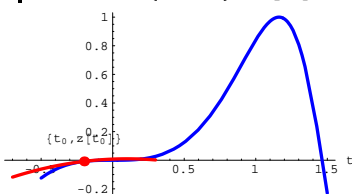
Another selection of t_0 :

```
t0 = 1.3;
{a, b} = {x[t0], y[t0]};
newcurve = Plot[znew[t], {t, t0 - 0.5, t0 + 0.5},
  PlotStyle -> {{Red, Thickness[0.01]}}, DisplayFunction -> Identity];
Show[actualcurve, newcurve, point[t0], PlotRange -> All,
  AxesLabel -> {"t", ""}, DisplayFunction -> $DisplayFunction];
```



Another selection of t_0 :

```
t0 = -0.2;
{a, b} = {x[t0], y[t0]};
newcurve = Plot[znew[t], {t, t0 - 0.5, t0 + 0.5},
  PlotStyle -> {{Red, Thickness[0.01]}}, DisplayFunction -> Identity];
Show[actualcurve, newcurve, point[t0], PlotRange -> All,
  AxesLabel -> {"t", ""}, DisplayFunction -> $DisplayFunction];
```

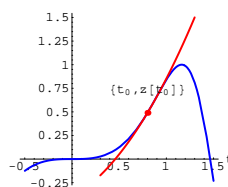


Play with some more selections of t_0 . Describe what you see and explain why you are seeing it.

□ Answer:

Do it again:

```
t0 = 0.8;
{a, b} = {x[t0], y[t0]};
newcurve = Plot[znew[t], {t, t0 - 0.5, t0 + 0.5},
  PlotStyle -> {{Red, Thickness[0.01]}}, DisplayFunction -> Identity];
Show[actualcurve, newcurve, point[t0], PlotRange -> All,
  AxesLabel -> {"t", ""}, DisplayFunction -> $DisplayFunction];
```



The two curves are tangent to each other at their point of contact at $\{t_0, z[t_0]\}$. When you think about it, you'll have to agree this outcome is, as Lewis Carroll put it, "large as life and twice as natural."

Reason:

One curve is

$$z[t] = f[x[t], y[t]];$$

the other is

$$z_{new}[t] = \text{linearf}[x[t], y[t], \{x[t_0], y[t_0]\}].$$

They are tangent at $\{t_0, z[t_0]\}$, as above, because the plane

$$z = \text{linearf}[x, y, \{x[t_0], y[t_0]\}]$$

is tangent to the surface

$$z = f[x, y]$$

at the point

$$\{x[t_0], y[t_0], f[x[t_0], y[t_0]]\} = \{x[t_0], y[t_0], z[t_0]\}.$$

And that's all there is to it!

This will work for any choice of $f[x, y]$, $x[t]$, and $y[t]$ you want.

□ B.4.b.i) Why the chain rule works

Given

$$z[t] = f[x[t], y[t]],$$

Mathematica calculates $z'[t_0]$ as follows:

```
Clear[x, y, z, t, f];
z[t_] = f[x[t], y[t]];
D[z[t], t] /. t -> t0
y'[t0] f^(0,1)[x[t0], y[t0]] + x'[t0] f^(1,0)[x[t0], y[t0]]
```

Most folks call this formula the "chain rule."

Explain where the chain rule comes from.

□ Answer:

As you saw above, the two curves

$$z[t] = f[x[t], y[t]]$$

and

$$z_{new}[t] = \text{linearf}[x[t], y[t], \{x[t_0], y[t_0]\}]$$

are tangent at $\{t_0, z[t_0]\}$.

This tells you that $z'[t_0] = z_{new}'[t_0]$.

Look at $z_{new}[t]$ for cleared functions $f[x, y]$, $x[t]$, and $y[t]$:

```
Clear[a, b, x, y, z, gradf, linearf, znew];
gradf[x_, y_] = {D[f[x, y], x], D[f[x, y], y]};
linearf[x_, y_, {a_, b_}] := f[a, b] + gradf[a, b] . {x - a, y - b};
znew[t_] = linearf[x[t], y[t], {a, b}] /. {a -> x[t0], b -> y[t0]}
f[x[t0], y[t0]] + (y[t] - y[t0]) f^(0,1)[x[t0], y[t0]] +
(x[t] - x[t0]) f^(1,0)[x[t0], y[t0]]
```

You don't need Mathematica to see that $z_{new}'[t]$ is:

```
D[znew[t], t]
y'[t] f^(0,1)[x[t0], y[t0]] + x'[t] f^(1,0)[x[t0], y[t0]]
```

But

$$z'[t_0] = znew'[t_0];$$

so $z'[t_0]$ is given by:

```
D[znew[t], t] /. t -> t0
y'[t0] f^(0,1)[x[t0], y[t0]] + x'[t0] f^(1,0)[x[t0], y[t0]]
```

This explains Mathematica's calculation:

```
z[t_] = f[x[t], y[t]];
D[z[t], t] /. t -> t0
y'[t0] f^(0,1)[x[t0], y[t0]] + x'[t0] f^(1,0)[x[t0], y[t0]]
D[f[x[t], y[t]], t] /. t -> t0
y'[t0] f^(0,1)[x[t0], y[t0]] + x'[t0] f^(1,0)[x[t0], y[t0]]
```

This explains where the chain rule comes from.

□B.4.b.ii) Why the chain rule works for functions of three variables

Given

$$w[t] = f[x[t], y[t], z[t]],$$

Mathematica calculates $w'[t_0]$ as follows:

```
Clear[x, y, z, w, t, f, t0]
w[t_] = f[x[t], y[t], z[t]];
D[w[t], t] /. t -> t0
z'[t0] f^(0,0,1)[x[t0], y[t0], z[t0]] + y'[t0] f^(0,1,0)[x[t0], y[t0], z[t0]] +
x'[t0] f^(1,0,0)[x[t0], y[t0], z[t0]]
```

This is the chain rule for functions of three variables.

Explain where it comes from.

□Answer:

You just linearize

$$f[x, y, z]$$

at a point $\{a, b, c\}$ with

$$\text{linear}[x, y, z, \{a, b, c\}] = f[a, b, c] + \text{grad}[a, b, c] \cdot \{x - a, y - b, z - c\}$$

and proceed as above.

This time you will not be able to do the plots, but the calculations will

hold up.

VC.03 The Gradient Tutorials

T.1) The total differential

If you are given a function $f[x]$ of one variable, then you can write the differential

$$df = f'[x] dx.$$

This is really just a suggestive notation, and it is handy for hand calculations:

→ You divide both sides of

$$df = f'[x] dx$$

by dx to get

$$\frac{df}{dx} = f'[x]$$

Interpret this as

$$\frac{d(f[x(t)])}{dt} = f'[x(t)] \frac{dx(t)}{dt} = f'[x(t)] x'[t].$$

This is the chain rule for functions of one variable.

□T.1.a.i)

How do you carry off this stunt for a function $f[x, y]$?

□Answer:

Very easily.

Just write

$$df = D[f[x, y], x] dx + D[f[x, y], y] dy.$$

Divide both sides by dt to get

$$\frac{df}{dt} = D[f[x, y], x] \frac{dx}{dt} + D[f[x, y], y] \frac{dy}{dt}.$$

This gives you the chain rule:

```
Clear[f, x, y, t]
D[f[x[t], y[t]], t]
y'[t] f^(0,1)[x[t], y[t]] + x'[t] f^(1,0)[x[t], y[t]]
```

Remember

$$D[f[x, y], x] = f^{(1,0)}[x, y]$$

and

$$D[f[x, y], y] = f^{(0,1)}[x, y].$$

It pays to notice once again that

$$\frac{df}{dt} = D[f[x, y], x] \frac{dx}{dt} + D[f[x, y], y] \frac{dy}{dt}$$

is nothing more than

$$\text{grad}[x, y] \cdot \{x'[t], y'[t]\},$$

which is the chain rule.

□T.1.a.ii)

Use the total differential

$$df = D[f[x, y], x] dx + D[f[x, y], y] dy$$

to help do a hand calculation of

$$D[\text{Sin}[x[t]^2 y[t]], t].$$

□Answer by hand calculation:

Take $f[x, y] = \text{Sin}[x^2 y]$.

So

$$df = (\text{Cos}[x^2 y] 2 x y) dx + (\text{Cos}[x^2 y] x^2) dy.$$

Divide both sides by dt to get:

$$\frac{df}{dt} = (\text{Cos}[x^2 y] 2 x y) \frac{dx}{dt} + (\text{Cos}[x^2 y] x^2) \frac{dy}{dt}.$$

Interpret the result as

$$\frac{d(\text{Sin}[x[t]^2 y[t]])}{dt} = (\text{Cos}[x[t]^2 y[t]] 2 x[t] y[t]) \frac{dx[t]}{dt} + (\text{Cos}[x[t]^2 y[t]] x[t]^2) \frac{dy[t]}{dt}$$

This is in agreement with:

```
Clear[x, y, t]
Expand[D[Sin[x[t]^2 y[t]], t]]
2 Cos[x[t]^2 y[t]] x[t] y[t] x'[t] + Cos[x[t]^2 y[t]] x[t]^2 y'[t]
```

This is a nice hand technique that's especially handy when you are working with a pencil on the back of an old envelope.

□T.1.b)

Use the total differential to give a hand derivation of the product rule

$$D[x[t] y[t] z[t], t] = x'[t] y[t] z[t] + x[t] y'[t] z[t] + x[t] y[t] z'[t]$$

□Answer by hand:

Put

$$f[x, y, z] = x y z.$$

This gives

$$df = y z dx + x z dy + x y dz.$$

Divide both sides by dt to get

$$\frac{df}{dt} = y z \frac{dx}{dt} + x z \frac{dy}{dt} + x y \frac{dz}{dt}.$$

Interpret this as

$$\begin{aligned} \frac{d(x[t] y[t] z[t])}{dt} &= y[t] z[t] x'[t] + x[t] z[t] y'[t] + x[t] y[t] z'[t] \\ &= x'[t] y[t] z[t] + x[t] y'[t] z[t] + x[t] y[t] z'[t]. \end{aligned}$$

Check:

```
Clear[x, y, z, t]
D[x[t] y[t] z[t], t]
y[t] z[t] x'[t] + x[t] z[t] y'[t] + x[t] y[t] z'[t]
```

Yep.

□T.1.c)

Use the total differential to help give a formula for the derivative with respect to t of

$$g[t] = \int_{\cos[t]}^{\sin[t]} h[s] ds$$

where $h[s]$ is an unspecified function.

□Answer:

Put

$$f[x, y] = \int_y^x h[s] ds.$$

The total differential is

$$\begin{aligned} df &= D[f[x, y], x] dx + D[f[x, y], y] dy \\ &= h[x] dx - h[y] dy. \end{aligned}$$

Divide through by dt to get

$$\frac{df}{dt} = h[x] \frac{dx}{dt} - h[y] \frac{dy}{dt}.$$

Now put

$$x = \sin[t] \text{ and } y = \cos[t]$$

to get

$$\begin{aligned} D\left[\int_{\cos[t]}^{\sin[t]} h[s] ds, t\right] &= h[\sin[t]] \cos[t] - h[\cos[t]] (-\sin[t]) \\ &= h[\sin[t]] \cos[t] + h[\cos[t]] \sin[t]. \end{aligned}$$

Would you have gotten it right without using the total differential?

Probably not.

T.2) What's the chain rule good for?**□T.2.a)**

Is the chain rule formula for functions of more than one variable useful for calculating derivatives of specific functions?

□Answer:

Not really.

In specific situations, the chain rule for more than one variable is not really needed because if you make the substitutions first and then differentiate, then all you need is the chain rule for functions of one variable.

Case in point:

If you want to differentiate

$$\cos[x[t] y[t]^3]$$

with respect to t , you can use the chain rule for a function of one variable and the product rule (from the early part of the course) to get

$$\begin{aligned} \frac{d \cos[x[t] y[t]^3]}{dt} &= -\sin[x[t] y[t]^3] \frac{d(x[t] y[t]^3)}{dt} \\ &= -\sin[x[t] y[t]^3] (x[t] 3 y[t]^2 y'[t] + x'[t] y[t]^3). \end{aligned}$$

You could also use the two variable chain rule from this lesson. But the point is that you don't really need the two variable chain rule to do this problem.

□T.2.b)

If the chain rule formula for functions of more than one variable is not particularly useful for calculating derivatives of specific functions, then what the heck is it good for?

□Answer:

Instead of being a great calculational tool, the chain rule for functions of more than one variable is a great theoretical tool.

Reason:

The chain rule for functions of more than one variable helps you unlock the magic of the gradient. Cases in point:

→ The chain rule gave you the basis for explaining why the gradient is perpendicular to level curves and surfaces.

→ Once you knew this, it was not a big jump to see why the gradient points in the direction of greatest initial increase.

Should you forget about the chain rule?

Certainly not.

T.3) The gradient and maximization and minimization:**The FindMinimum instruction**

When you plot a surface, your eyes look for crest and dips. Tops of crests are the tops of the hills. Bottoms of dips are the deepest points in depressions that could collect rain water.

Fancy folks call the tops of crests by the name "local maxima." The same folks call the bottoms of the dips by the name "local minima." Most folks, fancy and down-to-earth, are interested in the maximum which is located at the top of the very highest crest. The same folks are also interested in the minimum which is located at the bottom of the deepest dip.

□T.3.a.i) Tops of the crests

How do you know that $\{x_0, y_0, f[x_0, y_0]\}$ sits at the top of a crest on the surface $z = f[x, y]$, then $\text{grad}f[x_0, y_0] = \{0, 0\}$?

□Answer:

If you are at $\{x_0, y_0\}$ and you leave $\{x_0, y_0\}$ in the direction of $\text{grad}f[x_0, y_0]$, then $f[x, y]$ initially goes UP unless

$$\text{grad}f[x_0, y_0] = \{0, 0\}.$$

The upshot:

If $\text{grad}f[x_0, y_0]$ is not $\{0, 0\}$, then $\{x_0, y_0, f[x_0, y_0]\}$ cannot sit at the top of a crest on the surface $z = f[x, y]$. In other words, if

$\{x_0, y_0, f[x_0, y_0]\}$ sits at the top of a crest on the surface $z = f[x, y]$, then

$$\text{grad}f[x_0, y_0] = \{0, 0\}.$$

□T.3.a.ii) Bottoms of the dips

How do you know that $\{x_0, y_0, f[x_0, y_0]\}$ sits at the bottom of a dip on the surface $z = f[x, y]$, then $\text{grad}f[x_0, y_0] = \{0, 0\}$?

□Answer:

If you are at $\{x_0, y_0\}$ and you leave $\{x_0, y_0\}$ in the direction of $-\text{grad}f[x_0, y_0]$, then $f[x, y]$ initially goes DOWN unless

$$\text{grad}f[x_0, y_0] = \{0, 0\}.$$

The upshot:

If $\text{grad}f[x_0, y_0]$ is not $\{0, 0\}$, then $\{x_0, y_0, f[x_0, y_0]\}$ cannot sit at the bottom of a dip the surface $z = f[x, y]$. In other words, if

$\{x_0, y_0, f[x_0, y_0]\}$ sits at the bottom of a dip on the surface $z = f[x, y]$, then

$$\text{grad}f[x_0, y_0] = \{0, 0\}.$$

□T.3.b.i) Setting the gradient equal to 0

Find the maximizers and minimizers (if any) of
 $f[x, y] = 2.1x^2 + 5.9y^2 - 7.2x + 3.3y + 8.7$.

□Answer:

Look at the formula

$$f[x, y] = 2.1x^2 + 5.9y^2 - 7.2x + 3.3y + 8.7.$$

For large $|x|$ and $|y|$, the dominant terms

$$2.1x^2 + 5.9y^2$$

make $f[x, y]$ really huge. This means $f[x, y]$ has no maximum value.

It also means that there is no way for $f[x, y]$ to ever get near $-\infty$. So

$f[x, y]$ has a minimum (at the bottom of the deepest dip) and the

minimizer must be a point at which $\text{grad}f[x, y] = 0$:

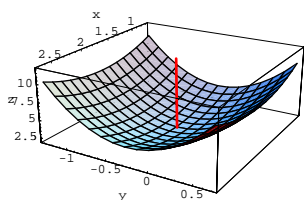
```
Clear[x, y, f, gradf]
f[x_, y_] = 2.1 x^2 + 5.9 y^2 - 7.2 x + 3.3 y + 8.7;
gradf[x_, y_] = {D[f[x, y], x], D[f[x, y], y]};
Solve[gradf[x, y] == {0, 0}, {x, y}]
{{x -> 1.71429, y -> -0.279661}}
```

The minimizer is $\{1.71429, -0.279661\}$ and the minimum value is $f[1.71429, -0.279661]$:

```
f@@{1.71429, -0.279661}
2.06713
```

Take a look:

```
deepestdip = {1.71429, -0.279661, 2.06713};
flagpole = Graphics3D[
  {Red, Thickness[0.01], Line[{deepestdip, {1.7, -0.3, 12}}]}];
surface = Plot3D[f[x, y], {x, 1.7 - 1, 1.7 + 1},
  {y, -0.3 - 1, -0.3 + 1}, DisplayFunction -> Identity];
Show[surface, flagpole, AxesLabel -> {"x", "y", "z"},
  ViewPoint -> CMView, DisplayFunction -> $DisplayFunction];
```



The flagpole is planted at the bottom of the deepest dip.

□T.3.b.ii) More than one candidate

Find the maximizers and minimizers (if any) of
 $f[x, y] = 0.2x^4 + 0.1y^4 + 8.2xy - 18.4x$

□Answer:

Look at the formula

$$f[x, y] = 0.2x^4 + 0.1y^4 + 8.2xy - 18.4x$$

For large $|x|$ and $|y|$, the dominant terms $0.2x^4 + 0.1y^4$ make $f[x, y]$ really huge. This means $f[x, y]$ has no maximum. It also means that

there is no way for $f[x, y]$ to ever get near $-\infty$; so $f[x, y]$ has a

minimum and the minimizer must be a point at which

$$\text{grad}f[x, y] = 0:$$

```
Clear[x, y, f, gradf]
f[x_, y_] = 0.2 x^4 + 0.1 y^4 + 8.2 x y - 18.4 x;
gradf[x_, y_] = {D[f[x, y], x], D[f[x, y], y]};
Solve[gradf[x, y] == {0, 0}, {x, y}]
{{x -> -2.53906 - 3.06993 I, y -> -3.16285 + 2.96988 I},
 {x -> -2.53906 + 3.06993 I, y -> -3.16285 - 2.96988 I},
 {x -> -2.45864, y -> 3.69388}, {x -> -0.653039 - 3.63228 I,
 y -> -0.250639 - 4.22198 I}, {x -> -0.653039 + 3.63228 I,
 y -> -0.250639 + 4.22198 I}, {x -> -0.564143, y -> 2.26142},
 {x -> 2.66448 - 1.68863 I, y -> 2.62211 + 3.03901 I},
 {x -> 2.66448 + 1.68863 I, y -> 2.62211 - 3.03901 I},
 {x -> 4.07802, y -> -4.37255}}
```

Tossing out the complex candidates gives three candidates for the minimizer:

```
candidate1 = {4.07802, -4.37255};
candidate2 = {-0.564143, 2.26142};
candidate3 = {-2.45864, 3.69388};
```

Compare:

```
{f@@candidate1, f@@candidate2, f@@candidate3}
{-129.385, 2.55454, -3.30667}
```

It's no contest. The minimizer is:

```
candidate1
{4.07802, -4.37255}
```

The minimum value is:

```
f@@candidate1
-129.385
```

Not much to it.

□T.3.c.i) Terrible gradients and the FindMinimum instruction

Find the maximizers and minimizers (if any) of

$$f[x, y] = 1.2e^{x^2} + 2.9e^{y^2} - 8.2xy^2.$$

Show off your results with a plot.

□Answer:

Look at the formula

$$f[x, y] = 1.2e^{x^2} + 2.9e^{y^2} - 8.2xy^2.$$

For large $|x|$ and $|y|$, the dominant terms

$$1.2e^{x^2} + 2.9e^{y^2}$$

make $f[x, y]$ hugely positive. This means $f[x, y]$ has no maximum. It

also means that there is no way for $f[x, y]$ to ever get near $-\infty$; so

$f[x, y]$ has a minimum value and the minimizer must be a point at

which $\text{grad}f[x, y] = 0$:

```
Clear[f, gradf, x, y]
f[x_, y_] = 1.2 E^{x^2} + 2.9 E^{y^2} - 8.2 x y^2;
gradf[x_, y_] = {D[f[x, y], x], D[f[x, y], y]};
{2.4 E^{x^2} x - 8.2 y^2, 5.8 E^{y^2} y - 16.4 x y}
Solve[gradf[x, y] == {0, 0}, {x, y}]
Solve[{2.4 E^{x^2} x - 8.2 y^2, 5.8 E^{y^2} y - 16.4 x y} == {0, 0}, {x, y}]
```

Dung.

Mathematica had to bail out.

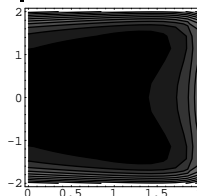
But this is no reason for you to give up. Take another look at the formula:

$$f[x, y] = 1.2e^{x^2} + 2.9e^{y^2} - 8.2xy^2$$

For the smaller values of $|x|$ and $|y|$, the term $8.2xy^2$ has a lot of influence. Also, to drive $f[x, y]$ down, you'll want $x > 0$.

Look at plot of some level curves of $f[x, y]$ for the smaller values of $|x|$ and $|y|$ with $x > 0$:

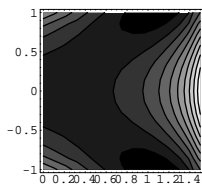
```
ContourPlot[f[x, y], {x, 0, 2}, {y, -2, 2}];
```



The lighter the shading, the higher the function.

The minimizer(s) lurk inside dark zone. Look again:

```
ContourPlot[f[x, y], {x, 0, 1.5}, {y, -1, 1}];
```



This is evidence pointing to two minimizers. Look at the formula:

```
f[x, y]
1.2 E^x^2 + 2.9 E^y^2 - 8.2 x y^2
```

And look at:

```
f[x, -y]
1.2 E^x^2 + 2.9 E^y^2 - 8.2 x y^2
```

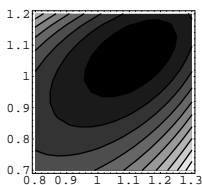
Aha!

$$f[x, y] = f[x, -y].$$

This tells you that if you locate one minimizer at $\{x^*, y^*\}$, then $\{x^*, -y^*\}$ is also a minimizer.

Look high:

```
ContourPlot[f[x, y], {x, 0.8, 1.3}, {y, 0.7, 1.2}];
```



You've got one of the minimizers trapped!

It's somewhere in the dark zone near $\{1.05, 1.1\}$. Now move in for the

kill with the Mathematica instruction FindMinimum starting with an initial guess of $\{1.05, 1.1\}$:

```
minimizer = FindMinimum[f[x, y], {x, 1.05}, {y, 1.1}]
{2.80399, {x -> 1.12131, y -> 1.07421}}
```

Mathematica is telling you that it has located a bottom of a dip at the point

```
{1.12131, 1.07421, 2.80399}.
```

Test the gradient at $\{1.12131, 1.07421\}$:

```
gradf@@{1.12131, 1.07421}
{-0.0000252787, 0.000119753}
```

Great; it's darn close to $\{0, 0\}$.

Now you can say with considerable authority that your estimate is that the minimizers are

```
{1.12131, 1.07421} and {1.12131, -1.07421}
```

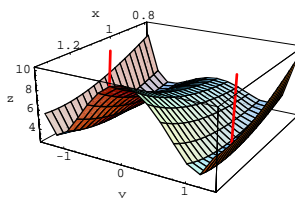
and the minimum value of $f[x, y]$ is:

```
f@@{1.12131, 1.07421}
2.80399
f@@{1.12131, -1.07421}
2.80399
```

Notice that this is the same as the first slot of the output of the FindMinimum instruction.

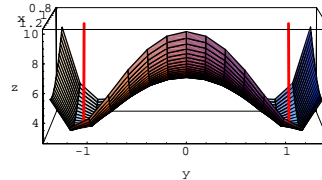
Take a look at the fruits of your labor:

```
h = 0.3;
deepestdip1 = {1.12131, 1.07421, 2.80399};
deepestdip2 = {1.12131, -1.07421, 2.80399};
flagpoles = {Graphics3D[{Red, Thickness[0.01],
  Line[{deepestdip1, {1.1, 1.07, 10}}]}, Graphics3D[
  {Red, Thickness[0.01], Line[{deepestdip2, {1.1, -1.07, 10}}]}];];
surface =
Plot3D[f[x, y], {x, 1.1 - h, 1.1 + 0.8 h}, {y, -1.07 - h, 1.07 + h},
  PlotPoints -> {20, 15}, DisplayFunction -> Identity];
Show[surface, flagpoles, AxesLabel -> {"x", "y", "z"},
  ViewPoint -> CMView, DisplayFunction -> $DisplayFunction];
```



The two flagpoles are planted at the bottoms of the two dips which are equally deep. Here's a look from a different viewpoint:

```
Show[surface, flagpoles, AxesLabel -> {"x", "y", "z"},
  ViewPoint -> {10, 0, 1}, DisplayFunction -> $DisplayFunction];
```



□T.3.c.ii) How FindMinimum works

Roughly speaking, how did the FindMinimum instruction find the minimizer in part i)?

□Answer:

Here're the function and its gradient.

```
Clear[f, gradf, x, y]
f[x_, y_] = 1.2 E^x^2 + 2.9 E^y^2 - 8.2 x y;
gradf[x_, y_] = {D[f[x, y], x], D[f[x, y], y]}
{2.4 E^x^2 x - 8.2 y, -8.2 x + 5.8 E^y^2 y}
```

Starting with an initial guess at $\{1.05, 1.1\}$, the FindMinimum instruction found a minimizer at:

```
minimizer = {1.12131, 1.07421}
{1.12131, 1.07421}
```

To get to this point, Mathematica tried to move from $\{1.05, 1.1\}$ on a path whose tangent vectors always point with the negative gradient of $f[x, y]$.

This is a good strategy because when you go in the direction of the negative gradient, you drive $f[x, y]$ down.

Fancy folks call this by the name "steepest descent."

□T.3.c.iii)

Can you use FindMinimum to help to find maximizers?

□Answer:

Sure.

The maximizers of $f[x, y]$ are the minimizers of $-f[x, y]$.

You can use FindMinimum[-f[x, y], {x, a}, {y, b}] to start your search at $\{a, b\}$ for maximizers of $f[x, y]$. This time Mathematica will try to start at $\{a, b\}$ and go on a path whose tangent vectors always point with the positive gradient of $f[x, y]$.

□T.3.c.iv) Pretenders versus the real thing

Is there a trick to using FindMinimum?

□Answer:

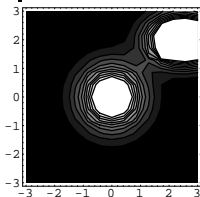
Sometimes using FindMinimum is as tricky as one of our former presidents who had the nickname "Tricky Dick."

To see what can happen, go with

$$f[x, y] = 4 e^{-(x^2 + y^2)} + 8 e^{-((x-2.5)^2 + 2(y-2)^2)}$$

This function is never negative or zero, but as $|x|$ and $|y|$ get big, $f[x, y]$ gets really close to 0. Consequently $f[x, y]$ has no minimum value but has at least one maximizer. Look at a plot of some level curves:

```
Clear[f, gradf, x, y]
f[x_, y_] = 4 E^-(x^2+y^2) + 8 E^-(x-2.5)^2+2 (y-2)^2;
ContourPlot[f[x, y], {x, -3, 3}, {y, -3, 3}];
```



The lighter the shading, the higher $f[x, y]$ is.

Crests have been trapped near $\{0, 0\}$ and $\{2, 2\}$.

Search for the maximizer at starting at $\{2, 2\}$:

```
FindMinimum[-f[x, y], {x, 2}, {y, 2}]
{-8.00014, {x → 2.49996, y → 1.99998}}
candidate1 = {2.49996, 1.99998}
{2.49996, 1.99998}
```

Test it:

```
Clear[gradf]
gradf[x_, y_] = {D[f[x, y], x], D[f[x, y], y]};
gradf@@candidate1
{-0.0000673367, 0.0000741272}
```

Fairly good.

Search for the maximizer at starting at $\{0, 0\}$:

```
FindMinimum[-f[x, y], {x, 0}, {y, 0}]
{-4.00001, {x → 3.23816 × 10^-6, y → 5.18105 × 10^-6}}
candidate2 = {3.23856 / 10^6, 5.18174 / 10^6}
{3.23856 × 10^-6, 5.18174 × 10^-6}
```

Test it:

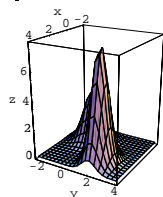
```
gradf@@candidate2
{-3.21154 × 10^-9, -5.54416 × 10^-9}
```

Good.

Two maximizers?

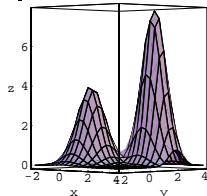
Take a look at a plot:

```
surface = ParametricPlot3D[{x, y, f[x, y]}, {x, -2, 4}, {y, -2, 4},
ViewPoint → CMView, PlotRange → All, AxesLabel → {"x", "y", "z"}];
```



Take a look from a different viewpoint:

```
Show[surface, ViewPoint → {8, -8, 0}];
```



There are two crests. One exhibits the true maximum value; the other is just a silly pretender sitting at the top of its own short crest. The true maximizer sits on the top of the highest crest, master of all.

Fancy dudes call the pretender a "relative maximizer" or a "local maximizer." The same fancy dudes call the true maximizers or minimizers "global maximizers" or "global minimizers."

Compare:

```
{f@@candidate1, f@@candidate2}
{8.00014, 4.00001}
```

The maximizer is:

```
candidate1
{2.49996, 1.99998}
```

The maximum value is 8.00014.

The trick to using `FindMinimum[g[x, y], {x, a}, {y, b}]` is to pick a starting point $\{a, b\}$ close enough to the true minimizer of $g[x, y]$.

Sometimes this isn't as easy you might wish.

□ T.3.c.v)

Can you use `FindMinimum` for functions of more than two variables?

□ Answer:

Yep.

T.4) Eye-balling a function for max-min

□ T.4.a)

Does

$$f[x, y] = \frac{x^4}{10} - 3xy + \frac{y^4}{7} - 8x + 9y$$

have a maximizer or a minimizer?

□ Answer:

Look at the formula

$$f[x, y] = \frac{x^4}{10} - 3xy + \frac{y^4}{7} - 8x + 9y.$$

When $|x|$ and $|y|$ are large then the dominant terms

$$\frac{x^4}{10} + \frac{y^4}{7}$$

make $f[x, y]$ really huge, so the global scale plot of $f[x, y]$ looks like a cup. This means $f[x, y]$ has no tallest crest but does have a deepest dip.

As a result, $f[x, y]$ has no maximum value but does have a minimum value.

□ T.4.b)

Does

$$f[x, y] = e^{-(x^2-y^2)} (x^6 + 7y^2 + 4)$$

have a maximizer or a minimizer?

□ Answer:

Look at the formula

$$f[x, y] = e^{-(x^2-y^2)} (x^6 + 7y^2 + 4).$$

When $|x|$ and $|y|$ are large, then $f[x, y]$ is incredibly close to 0.

Also $f[x, y] > 0$ for all x 's and y 's. As a result this the surface $z = f[x, y]$ has a biggest crest, but no dip below 0.

Consequently $f[x, y]$ has a maximizer but no minimizer.

□ T.4.c)

Does

$$f[x, y] = \frac{x^3 + 7y^2}{1 + x^4 + y^6}$$

have a maximizer or a minimizer?

□ Answer:

Look at the formula

$$f[x, y] = \frac{x^3 + 7y^2}{1 + x^4 + y^6}.$$

When $|x|$ and $|y|$ are large, then the dominant terms in the denominator make $f[x, y]$ incredibly close to 0.

Also $f[x, 0] > 0$ for $x > 0$ and $f[x, 0] < 0$ for $x < 0$.

Now you know that $f[x, y]$ has positive and negative values and $f[x, y]$ is arbitrarily close to 0 as $|x|$ and $|y|$ become large. Consequently the surface $z = f[x, y]$ has a biggest crest and a deepest dip. This means $f[x, y]$ has both a maximizer and a minimizer.

T.5) Data fit

One of the central uses of minimization is to try to fit data by curves. Usually this involves the idea of least squares, an idea you have seen earlier in the course. In fact, a lot of this problem might be old hat to some of you.

□T.5.a.i)

Given a list of data points

$$\{\{a_1, b_1\}, \{a_2, b_2\}, \{a_3, b_3\}, \dots, \{a_n, b_n\}\},$$

you reach into your box of functions, pick a few, say $f[x]$, $g[x]$, and $h[x]$ and then you form the square error function

$$\begin{aligned} \text{sqerror}[r, s, t] = & (r f[a_1] + s g[a_1] + t h[a_1] - b_1)^2 \\ & + (r f[a_2] + s g[a_2] + t h[a_2] - b_2)^2 \\ & + (r f[a_3] + s g[a_3] + t h[a_3] - b_3)^2 \\ & + \dots \\ & + (r f[a_n] + s g[a_n] + t h[a_n] - b_n)^2. \end{aligned}$$

Then you minimize $\text{sqerror}[r, s, t]$ and assess the resulting fit.

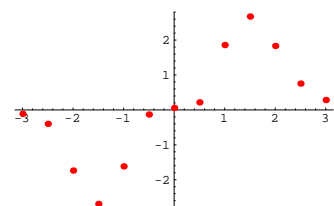
(As $|r|$, $|s|$, and $|t|$ get really large, $\text{sqerror}[r, s, t]$ also gets really large; so you are guaranteed that $\text{sqerror}[r, s, t]$ has a minimizer.)

Here are some data to work with:

```
data = {{-3., -0.1119}, {-2.5, -0.3986}, {-2., -1.7359},
{-1.5, -2.6926}, {-1., -1.6163}, {-0.5, -0.1308}, {0., 0.0570},
{0.5, 0.2171}, {1., 1.8582}, {1.5, 2.6742}, {2., 1.8313},
{2.5, 0.7550}, {3., 0.2831}}
{{-3., -0.1119}, {-2.5, -0.3986}, {-2., -1.7359}, {-1.5, -2.6926},
{-1., -1.6163}, {-0.5, -0.1308}, {0., 0.057}, {0.5, 0.2171},
{1., 1.8582}, {1.5, 2.6742}, {2., 1.8313}, {2.5, 0.755}, {3., 0.2831}}
```

And a plot:

```
dataplot = ListPlot[data, PlotStyle -> {PointSize[0.02], Red};
```



Try to find a function whose plot runs close to all these points.

□Answer:

Looks like points picked off the plot of an odd function plotted from $-\pi$ to π .

Fourier (1768-1830) revolutionized mathematics with the idea that all odd functions plotted on $[-\pi, \pi]$ are made up of combinations of $\text{Sin}[x]$, $\text{Sin}[2x]$, $\text{Sin}[3x]$,...

See whether Fourier was right.

Reach into the box of functions and pull out

$$\text{Sin}[x], \text{Sin}[2x], \text{and } \text{Sin}[3x].$$

Try to fit these data with a function of the form

$$r \text{Sin}[x] + s \text{Sin}[2x] + t \text{Sin}[3x],$$

where r , s , and t are to be selected to give the best possible fit.

```
Clear[fitter, sqerror, gradsqerror, s, t, r, x, k]
fitter[x_] = r Sin[x] + s Sin[2 x] + t Sin[3 x]
r Sin[x] + s Sin[2 x] + t Sin[3 x]
```

Now define the square error function and minimize it by setting its gradient equal to 0.

```
sqerror[r_, s_, t_] = Sum[(fitter[data[[k, 1]]] - data[[k, 2]])^2,
{ k, 1, Length[data] }];
gradsqerror[r_, s_, t_] = {D[sqerror[r, s, t], r],
D[sqerror[r, s, t], s], D[sqerror[r, s, t], t]};
Solve[gradsqerror[r, s, t] == {0, 0, 0}]
```

```
{r -> 1.97845, s -> 0.0475817, t -> -0.663794}
```

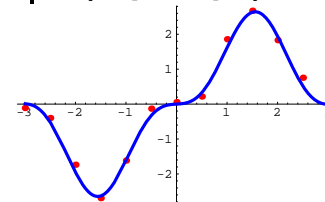
Here is the best you can do with

$\text{Sin}[x]$, $\text{Sin}[2x]$, and $\text{Sin}[3x]$:

```
Clear[bestfit]
bestfit[x_] = fitter[x] /. {r -> 1.97845, s -> 0.0475817, t -> -0.663794}
1.97845 Sin[x] + 0.0475817 Sin[2 x] - 0.663794 Sin[3 x]
```

Assess with a plot:

```
fitplot = Plot[bestfit[x], {x, -3, 3},
PlotStyle -> {{Thickness[0.01], RGBColor[0, 0, 1]}},
DisplayFunction -> Identity];
Show[dataplot, fitplot];
```



This is the best you can do with any function of the form:

```
fitter[x]
r Sin[x] + s Sin[2 x] + t Sin[3 x]
```

And this is not bad!

Maybe you could have done better by choosing different functions in place of $\text{Sin}[x]$, $\text{Sin}[2x]$, $\text{Sin}[3x]$.

Maybe you could have done better by choosing more functions in addition to $\text{Sin}[x]$, $\text{Sin}[2x]$, $\text{Sin}[3x]$.

Try it and see.

□T.5.a.ii) The Fit instruction

Do you have to endure the pain of all this typing every time you try to fit data?

□Answer:

This is a continuation of the last part. Please make sure that all instructions in the last part are alive on your machine.

No.

The Fit instruction in Mathematica will do this for you.

Here is what you did above done with the Fit instruction:

```
data = {{-3., -0.1119}, {-2.5, -0.3986},
{-2., -1.7359}, {-1.5, -2.6926}, {-1., -1.6163},
{-0.5, -0.1308}, {0., 0.0570}, {0.5, 0.2171}, {1., 1.8582},
{1.5, 2.6742}, {2., 1.8313}, {2.5, 0.7550}, {3., 0.2831}};
quickbestfit[x_] = Fit[data, {Sin[x], Sin[2 x], Sin[3 x]}, x]
1.97845 Sin[x] + 0.0475817 Sin[2 x] - 0.663794 Sin[3 x]
```

Compare with what you got in the last part:

```
bestfit[x]
1.97845 Sin[x] + 0.0475817 Sin[2 x] - 0.663794 Sin[3 x]
```

Exactly the same!

Thank you Stephen Wolfram and Mathematica.

□T.5.b.i) C&M students work with engineering students

The Calculus&Mathematica gang is doing a joint experiment with some engineers, not all of whom turn out to be the nerds you had expected. In this joint experiment, the group starts with a heated wire π units long with the temperature allowed to vary from position to position on the wire. As math people, the C&M gang thinks of the wire as the interval $0 \leq x \leq \pi$.

At the start of the experiment, the engineers instantly cool the ends at $x = 0$ and $x = \pi$ and maintain these ends at temperature 0.

At the same time, the engineers take pains to guarantee that the rest of the wire is perfectly insulated.

The engineers check their handbook and tell you that they have adjusted the time unit so that you should be able to get a good fit of the actual data by a function of the form

$$\text{temp}[x, t] = a_1 e^{-t} \sin[x] + a_2 e^{-4t} \sin[2x] + a_3 e^{-9t} \sin[3x] + \dots$$

where you can use as many terms as you need to get a good fit. The engineers carefully make some measurements on the wire reporting

$$\{x, t, \text{temp}[x, t]\}$$

for $x = 1, 1.5, 2, 2.5, 3$ and $t = 1, 2, 3$.

Here they are:

```
measurements = {{1., 0., 8.87}, {1., 1., 3.54},
{1., 2., 1.34}, {1., 3., 0.50}, {1.5, 0., 11.43}, {1.5, 1., 4.35},
{1.5, 2., 1.61}, {1.5, 3., 0.60}, {2., 0., 11.74}, {2., 1., 4.16},
{2., 2., 1.5}, {2., 3., 0.55}, {2.5, 0., 9.04}, {2.5, 1., 2.86},
{2.5, 2., 1.}, {2.5, 3., 0.36}, {3., 0., 2.57}, {3., 1., 0.69},
{3., 2., 0.24}, {3., 3., 0.09}};
```

Your part of the experiment is to try to fit these measurements by a function $\text{temp}[x, t]$ of the form given above.

Do it and assess the quality of the fit.

□Answer:

```
measurements =
{{1., 0., 8.87}, {1., 1., 3.54}, {1., 2., 1.34}, {1., 3., 0.50},
{1.5, 0., 11.43}, {1.5, 1., 4.35}, {1.5, 2., 1.61}, {1.5, 3., 0.60},
{2., 0., 11.74}, {2., 1., 4.16}, {2., 2., 1.5}, {2., 3., 0.55},
{2.5, 0., 9.04}, {2.5, 1., 2.86}, {2.5, 2., 1.}, {2.5, 3., 0.36},
{3., 0., 2.57}, {3., 1., 0.69}, {3., 2., 0.24}, {3., 3., 0.09}};
Clear[fittemp]
fittemp[x_, t_] = Fit[measurements, {E^-t Sin[x], E^-4 t Sin[2 x],
E^-9 t Sin[3 x], E^-16 t Sin[4 x], E^-25 t Sin[5 x], E^-36 t Sin[6 x]},
{x, t}]
12.0222 E^-t Sin[x] - 1.46502 E^-4 t Sin[2 x] + 0.482672 E^-9 t Sin[3 x] -
0.15374 E^-16 t Sin[4 x] + 0.138106 E^-25 t Sin[5 x] - 0.126032 E^-36 t Sin[6 x]
```

Check the fit:

Here are the entries in the $\{x, t\}$ slots of the table of measurements:

```
Table[{measurements[[k, 1]], measurements[[k, 2]],
{k, 1, Length[measurements]}}]
{{1., 0.}, {1., 1.}, {1., 2.}, {1., 3.}, {1.5, 0.}, {1.5, 1.}, {1.5, 2.},
{1.5, 3.}, {2., 0.}, {2., 1.}, {2., 2.}, {2., 3.}, {2.5, 0.},
{2.5, 1.}, {2.5, 2.}, {2.5, 3.}, {3., 0.}, {3., 1.}, {3., 2.}, {3., 3.}}
```

The discrepancies are:

```
Table[fittemp@@
{measurements[[k, 1]], measurements[[k, 2]]} - measurements[[k, 3]],
{k, 1, Length[measurements]}}]
{0.00144327, 0.157201, 0.0286502, 0.00365447, 0.00407858, 0.0577974,
0.0128831, -0.00295041, 0.00601153, -0.11814, -0.0201755,
-0.005733, 0.00506373, -0.187338, -0.0257976, -0.00177573,
0.00179077, -0.0583435, -0.0102564, -0.00553007}
```

Hot Dog!

The fit is good all the way!

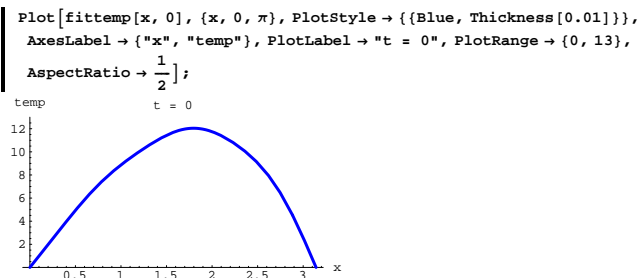
□T.5.b.ii)

The engineers ask for a plot displaying your estimate of the temperature of the wire x units from the left at time $t = 0$. And they ask for a plot displaying your estimate of the temperature of the wire x units from the left at time $t = 2$. Give them the plots.

□Answer:

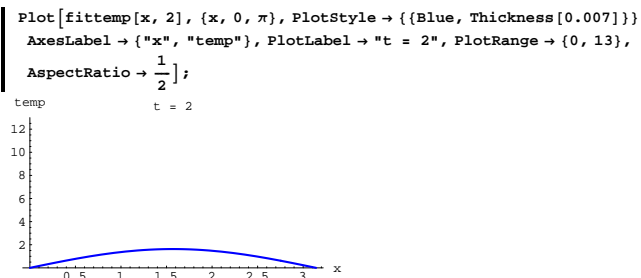
That's cake.

Here's the plot for $t = 0$:



When $t = 0$, the wire was hottest slightly to the right of center.

Here comes the plot for $t = 2$ with the same plot range as above:



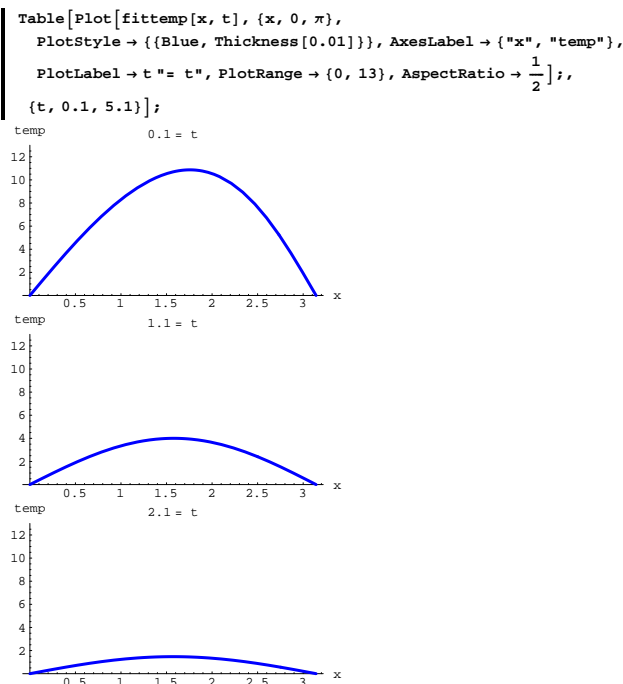
Cooled off quite a bit.

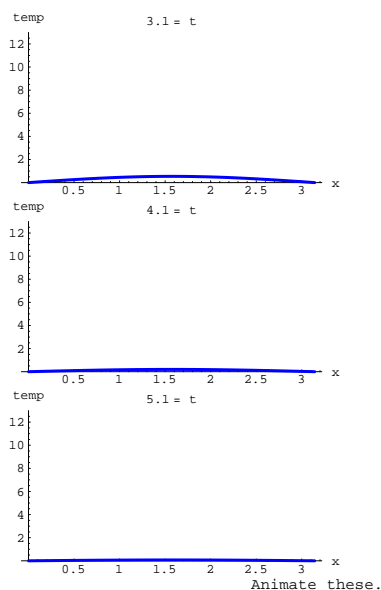
□T.5.b.iii)

The engineers ask for a movie that shows how the wire cools as t advances from 0.1 to 5.1.

□Answer:

No problem-o:

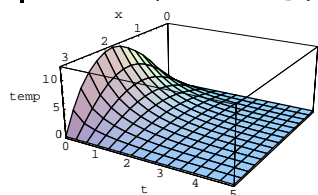




□T.5.b.iv)

One of the Calculus&Mathematica crew gets the idea to look at

```
Plot3D[fittemp[x, t], {x, 0, π}, {t, 0, 5},
  AxesLabel → {"x", "t", "temp"}, PlotRange → All, ViewPoint → CMView];
```



Interpret the plot.

□Answer:

You can see the initial temperature plot at the face where $t = 0$ and you can see that the entire wire is fairly cold by the time t gets to 3.

T.6) Lagrange's method for constrained maximization and minimization

Lagrange's method deals with maximizing or minimizing one function while another function is held constant. Once you are armed with the fact that the gradient gives you a normal vector, understanding it is pretty easy.

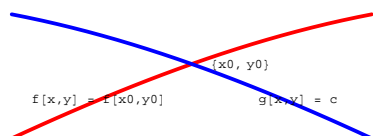
□T.6.a.i)

If $\{x_0, y_0\}$ is on the level curve $g[x, y] = c$ and $\{x_0, y_0\}$ maximizes or minimizes $f[x, y]$ given $g[x, y] = c$, then how do you know that the level curve $f[x, y] = f[x_0, y_0]$ is tangent to the level curve $g[x, y] = c$ at the point $\{x_0, y_0\}$?

□Answer:

Suppose $\{x_0, y_0\}$ is a point on the level curve $g[x, y] = c$ and the level curve $f[x, y] = f[x_0, y_0]$ crosses $g[x, y] = c$ without being tangent to $g[x, y] = c$. Like this:

Run the blank cell immediately below.



The first thing you can say is that $f[x, y]$ does not stay equal to $f[x_0, y_0]$ on the level curve $g[x, y] = c$.

This means that you can leave $\{x_0, y_0\}$ on the level curve $g[x, y] = c$ and go in one direction to lower the value of $f[x, y]$ and you can leave $\{x_0, y_0\}$ on the level curve $g[x, y] = c$ and go in the other direction to raise the value of $f[x, y]$.

Consequently:

If the level curve $f[x, y] = f[x_0, y_0]$ crosses $g[x, y] = c$ without being tangent to $g[x, y] = c$, then $f[x_0, y_0]$ is neither the maximum nor the minimum value of $f[x, y]$ given $g[x, y] = c$.

As a result, if $\{x_0, y_0\}$ maximizes or minimizes $f[x, y]$ given $g[x, y] = c$,

then level curve

$$f[x, y] = f[x_0, y_0]$$

is tangent to the level curve

$$g[x, y] = c \text{ at the point } \{x_0, y_0\}.$$

□T.6.a.ii)

If $\{x_0, y_0\}$ maximizes or minimizes $f[x, y]$ given $g[x, y] = c$, you know that the level curve $f[x, y] = f[x_0, y_0]$ is tangent to the level curve $g[x, y] = c$ at the point $\{x_0, y_0\}$.

Why does this force

$$\text{grad}f[x_0, y_0] = s \text{ grad}g[x_0, y_0]$$

for some number s , except in the case that $\text{grad}g[x_0, y_0] = 0$?

□Answer:

→ The tangent to the level curve $f[x, y] = f[x_0, y_0]$ must be parallel to the tangent to $g[x, y] = c$ at $\{x_0, y_0\}$.

So any normal vector to the level curve

$$f[x, y] = f[x_0, y_0] \text{ at } \{x_0, y_0\}$$

must be parallel to any normal vector of

$$g[x, y] = c \text{ at } \{x_0, y_0\}.$$

→ A normal vector to the level curve

$$f[x, y] = f[x_0, y_0] \text{ at } \{x_0, y_0\} \text{ is } \text{grad}f[x_0, y_0].$$

→ A normal vector to the level curve

$$g[x, y] = c \text{ at } \{x_0, y_0\} \text{ is } \text{grad}g[x_0, y_0].$$

Consequently $\text{grad}f[x_0, y_0]$ is parallel to $\text{grad}g[x_0, y_0]$.

A compact way of saying this is to say

$$\text{grad}f[x_0, y_0] = s \text{ grad}g[x_0, y_0]$$

for some number s except in the case $\text{grad}g[x_0, y_0] = 0$ in which case all bets are off.

The numbers s you get this way are called Lagrange multipliers by the heavy rollers.

Lagrange was a great French mathematician and astronomer who lived during the glory days 1736-1813.

□T.6.b.i)

What is Lagrange's method of maximizing or minimizing $f[x, y]$ given $g[x, y] = c$?

□Answer:

Lagrange's method of maximizing or minimizing

$f[x, y]$ given $g[x, y] = c$ is to solve the system of equations

$$\text{grad}f[x, y] = s \text{ grad}g[x, y]$$

and

$$g[x, y] = c$$

for $\{x, y\}$.

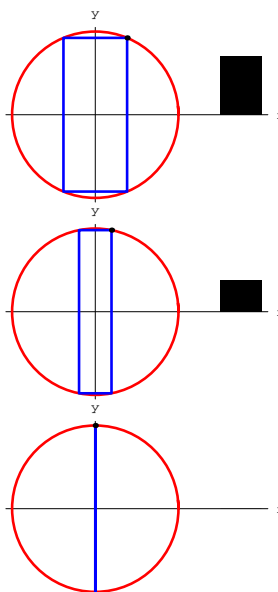
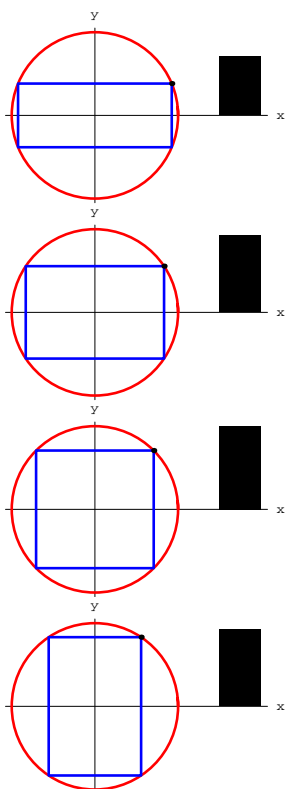
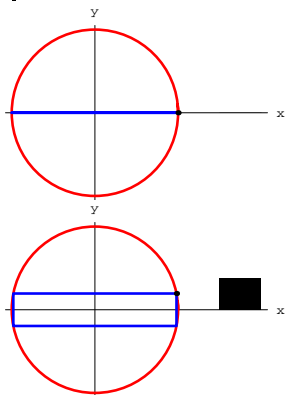
If you get all the $\{x, y\}$'s that solve this, then you are guaranteed that the true maximizers and minimizers are in this list.

□T.6.b.ii)

Look at the following movie:

The area of the bar graph on the right represents $\frac{1}{4}$ of the area of the inscribed rectangle.

```
circle = Graphics[{Red, Thickness[0.01], Circle[{0, 0}, 1]}];
Clear[rectangle, t]
rectangle[t_] = Graphics[
  {Thickness[0.01], Blue, Line[{{Cos[t], Sin[t]}, {-Cos[t], Sin[t]},
    {-Cos[t], -Sin[t]}, {Cos[t], -Sin[t]}, {Cos[t], Sin[t]}}]};
Clear[point]
point[t_] = Graphics[{PointSize[0.02], Point[{Cos[t], Sin[t]}]},
  Text[{"x", "y"}, {Cos[t] + .2, Sin[t]}];
Clear[areameter]
areameter[t_] = Graphics[Polygon[{{1.5, 0},
  {1.5, 2 Sin[t] Cos[t]}, {2, 2 Sin[t] Cos[t]}, {2, 0}, {1.5, 0}}]];
Table[Show[circle, rectangle[t], point[t],
  areameter[t], AspectRatio -> Automatic, Axes -> True,
  AxesOrigin -> {0, 0}, Ticks -> None, AxesLabel -> {"x", "y"}],
  {t, 0,  $\frac{\pi}{2}$ ,  $\frac{\pi}{16}$ }]
```



Animate these and run at a slow speed.

For a given prescribed radius r , use Lagrange's method to explain why the rectangle inscribed in the circle $x^2 + y^2 = r^2$ with corners at $\{x, y\}$, $\{x, -y\}$, $\{-x, -y\}$ and $\{-x, y\}$ with the biggest possible area is a square.

□Answer:

Saying that $\{x, y\}$ is on the circle of radius r centered at $\{0, 0\}$ is to say that

$$g[x, y] = x^2 + y^2 = r^2.$$

For a given $\{x, y\}$ on this circle, the area of the inscribed rectangle is

$$f[x, y] = 4xy.$$

To find the largest such rectangle, you've got to maximize $f[x, y]$ given $g[x, y] = r^2$.

This is a snap for Lagrange's method:

```
Clear[f, gradf, g, gradg, x, y, s, r]
f[x_, y_] = 4 x y;
g[x_, y_] = x^2 + y^2;
gradf[x_, y_] = {D[f[x, y], x], D[f[x, y], y]};
gradg[x_, y_] = {D[g[x, y], x], D[g[x, y], y]};
lagrange = Solve[{gradf[x, y] == s gradg[x, y], g[x, y] == r^2}, {x, y, s}]
{{s -> -2, x -> - $\frac{r}{\sqrt{2}}$ , y ->  $\frac{r}{\sqrt{2}}$ }, {s -> -2, x ->  $\frac{r}{\sqrt{2}}$ , y -> - $\frac{r}{\sqrt{2}}$ },
  {s -> 2, x -> - $\frac{r}{\sqrt{2}}$ , y -> - $\frac{r}{\sqrt{2}}$ }, {s -> 2, x ->  $\frac{r}{\sqrt{2}}$ , y ->  $\frac{r}{\sqrt{2}}$ }}
```

All four solutions for $\{x, y\}$ give a square; so the square is the largest inscribed rectangle.

This is not big news; you knew this all along.

What might be good news is the method. It was quick and easy.

□T.6.c.i)

Does Lagrange's method work finding the maximizers of $f[x, y, z]$ given $g[x, y, z] = c$?

Explain.

□Answer:

In theory, yes.

If $\{x_0, y_0, z_0\}$ is a point on the level surface $g[x, y, z] = c$ and the level surface $f[x, y, z] = f[x_0, y_0, z_0]$ cuts $g[x, y, z] = c$ at $\{x_0, y_0, z_0\}$ without being tangent to $g[x, y, z] = c$, then you can leave $\{x_0, y_0, z_0\}$ on the level surface $g[x, y, z] = c$ on one side of the level surface $f[x, y, z] = f[x_0, y_0, z_0]$ and raise $f[x, y, z]$.

And you can leave $\{x_0, y_0, z_0\}$ staying on the level surface $g[x, y, z] = c$ on the other side of $f[x, y, z] = f[x_0, y_0, z_0]$ and lower $f[x, y, z]$.

As a result if $\{x_0, y_0, z_0\}$ maximizes or minimizes $f[x, y, z]$ subject to $g[x, y, z] = c$, then the level surfaces

$f[x, y, z] = f[x_0, y_0, z_0]$ and $g[x, y, z] = c$
 must be tangent at $\{x_0, y_0, z_0\}$.
 So, as in the two variable case, their normals must be parallel at $\{x_0, y_0, z_0\}$.
 This means there is a number s such that
 $\text{grad}f[x_0, y_0, z_0] = s \text{grad}g[x_0, y_0, z_0]$
 except in the case $\text{grad}g[x_0, y_0, z_0] = 0$, in which case all bets are off.

□T.6.c.ii)

Calculus&Mathematica thanks Dr. Kevin Ford of the University of Illinois for helping out with this problem.

Use Lagrange's method to find the maximum and minimum values (if any) of $f[x, y, z] = xyz$ given $x + y + z = 6$ with $x > 0, y > 0, z > 0$.

□Answer:

The set up is to maximize and minimize

$$f[x, y, z] = x y z \text{ given } x + y + z = 6 \text{ with } x > 0, y > 0 \text{ and } z > 0.$$

You can push $f[x, y, z]$ close to (but not equal to) 0 while holding $x + y + z = 6$ by making x and y small and positive and setting

$$z = 6 - x - y.$$

For x and y small and positive, this makes $f[x, y, z]$ small and maintains the constraint $x + y + z = 6$:

```
Clear[x, y, z, f, g]
f[x_, y_, z_] = x y z;
g[x_, y_, z_] = x + y + z;
{f[x, y, 6 - x - y], g[x, y, 6 - x - y]} /. {x -> 0.01, y -> 0.01}
{0.000598, 6}
{f[x, y, 6 - x - y], g[x, y, 6 - x - y]} /. {x -> 0.00001, y -> 0.00002}
{1.19999 x 10^-9, 6}
```

Now you see that if $\{x, y, z\}$ is constrained by

$$g[x, y, z] = 6$$

with $x > 0, y > 0, z > 0$,

then $f[x, y, z]$ has no minimum value.

But maximizers are another story because there is no way to drive $f[x, y, z]$ to ∞ while maintaining the constraint

$$g[x, y, z] = x + y + z = 6$$

with $x > 0, y > 0$ and $z > 0$.

Here is how to use Lagrange's method to go after the maximizers of $f[x, y, z]$ subject to the constraint $g[x, y, z] = 6$ with $x > 0, y > 0$ and $z > 0$:

```
Clear[gradf, gradg, s]
gradf[x_, y_, z_] =
{D[f[x, y, z], x], D[f[x, y, z], y], D[f[x, y, z], z]};
gradg[x_, y_, z_] =
{D[g[x, y, z], x], D[g[x, y, z], y], D[g[x, y, z], z]};
Solve[{gradf[x, y, z] == s gradg[x, y, z], g[x, y, z] == 6}]
{{s -> 0, x -> 0, y -> 0, z -> 6}, {s -> 0, x -> 0, y -> 6, z -> 0},
{s -> 0, x -> 6, y -> 0, z -> 0}, {s -> 4, x -> 2, y -> 2, z -> 2}}
```

The candidates are:

```
Clear[candidate]
candidate[1] = {2, 2, 2}
candidate[2] = {6, 0, 0}
candidate[3] = {0, 6, 0}
candidate[4] = {0, 0, 6}
{2, 2, 2}
{6, 0, 0}
{0, 6, 0}
{0, 0, 6}
```

Compare:

```
Clear[k]
Table[{candidate[k], f@@candidate[k]}, {k, 1, 4}]
{{{2, 2, 2}, 8}, {{6, 0, 0}, 0}, {{0, 6, 0}, 0}, {{0, 0, 6}, 0}}
```

The maximizer of $f[x, y, z]$ subject to the constraint $g[x, y, z] = 6$ with $x > 0, y > 0$ and $z > 0$ is

$\{2, 2, 2\}$

and the constrained maximum value is 8:

```
f[2, 2, 2]
8
```

That fellow Lagrange was no jerk. You should like him because his work made it possible for you to take it easy.

□T.6.d)

Is there a catch?

□Answer:

Yes.

If the *Mathematica* solve command can't hack the algebra, then Lagrange's method is up the creek without a paddle.

VC.03 The Gradient Give it a Try!

Experience with the starred problems will be useful for understanding developments later in the course.

G.1) The gradient points in the direction of greatest initial increase*

□G.1.a)

Go with

$$f[x, y, z] = \text{Log}[3x^2 + y^2 + z^2].$$

In which direction should you leave the point $\{2.1, -3.7, 4.3\}$ to get the greatest possible initial increase of $f[x, y, z]$?

□G.1.b.i)

Set

$$f[x, y] = 2 \text{Sin}[0.3 x y]$$

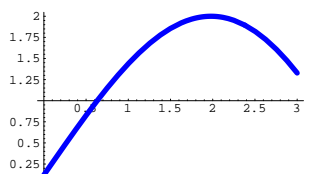
and look at a plot of $f[x[t], y[t]]$ for

$$\{x[t], y[t]\} = \{0.1, 2\} + t \text{grad}f[0.1, 2]$$

and $0 \leq t \leq 3$:

```
Clear[f, gradf, x, y, t]
f[x_, y_] = 2 Sin[0.3 x y];
gradf[x_, y_] = {D[f[x, y], x], D[f[x, y], y]};
{x[t_], y[t_]} = {0.1, 2} + t gradf[0.1, 2];

Plot[f[x[t], y[t]], {t, 0, 3},
PlotStyle -> {Blue, Thickness[0.02]}, PlotRange -> All,
AxesLabel -> {"t", ""}];
```



The knowledge of what fact about gradients gives you the ability to predict, before seeing the plot, that the curve must go up before it can go down?

Experiment with other functions $f[x, y]$.

Explain why the plot will go up initially no matter what function $f[x, y]$ you go with as long as $\text{grad}f[0.1, 2]$ is not $\{0, 0\}$.

□G.1.b.ii)

Set

$$f[x, y, z] = 0.6 \text{Cos}[1.8 x y z^2]$$

and look at a plot of

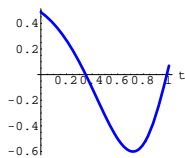
$$f[x[t], y[t], z[t]]$$

for

$$\{x[t], y[t], z[t]\} = \{0.3, -0.8, 1.2\} - t \text{grad}f[0.3, -0.8, 1.2]$$

and $0 \leq t \leq 1$:

```
Clear[f, gradf, x, y, z, t]
f[x_, y_, z_] = 0.6 Cos[1.8 x y z^2];
gradf[x_, y_, z_] =
{D[f[x, y, z], x], D[f[x, y, z], y], D[f[x, y, z], z]};
{x[t_], y[t_], z[t_]} = {0.3, -0.8, 1.2} - t gradf[0.3, -0.8, 1.2];
Plot[f[x[t], y[t], z[t]], {t, 0, 1},
PlotStyle -> {{Blue, Thickness[0.02]}}, AxesLabel -> {"t", ""}];
```



The knowledge of what fact about gradients gives you the ability to predict, before seeing the plot, that the curve must go DOWN before it can go up?

Experiment with other functions $f[x, y, z]$.

Explain why the plot will go down initially no matter what function $f[x, y, z]$ you go with as long as $\text{gradf}[0.3, -0.8, 1.2]$ is not $\{0, 0, 0\}$.

□G.1.c.i)

Here is the curve

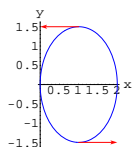
$$\{x[t], y[t]\} = \{\cos[t] + 1, \frac{3 \sin[t]}{2}\}$$

for $0 \leq t \leq 2\pi$, together with tangent vectors at the points

$$\{x[\frac{\pi}{2}], y[\frac{\pi}{2}]\} \text{ and } \{x[\frac{3\pi}{2}], y[\frac{3\pi}{2}]\}$$

all in true-scale:

```
Clear[x, y, t]
{x[t_], y[t_]} = {Cos[t] + 1, \frac{3 Sin[t]}{2}};
curveplot = ParametricPlot[{x[t], y[t]},
{t, 0, 2 \pi}, PlotStyle -> {{Blue, Thickness[0.01]}},
AxesLabel -> {"x", "y"}, DisplayFunction -> Identity];
Clear[tanvect]
tanvect[t_] :=
Arrow[{x'[t], y'[t]}, Tail -> {x[t], y[t]}, VectorColor -> Red];
setup = Show[curveplot, tanvect[\frac{\pi}{2}], tanvect[\frac{3\pi}{2}],
AspectRatio -> Automatic, DisplayFunction -> $DisplayFunction];
```



Now go with

$$f[x, y] = 4 - (x + 1)^2 - y^2$$

and add to the plot the gradient vector

$$\text{gradf}[x[\frac{\pi}{2}], y[\frac{\pi}{2}]]$$

with tail at

$$\{x[\frac{\pi}{2}], y[\frac{\pi}{2}]\}$$

and

$$\text{gradf}[x[\frac{3\pi}{2}], y[\frac{3\pi}{2}]]$$

with tail at

$$\{x[\frac{3\pi}{2}], y[\frac{3\pi}{2}]\}.$$

Remembering that

$$D[f[x[t], y[t]], t] = \text{gradf}[x[t], y[t]] \cdot \{x'[t], y'[t]\},$$

use the information displayed in your plot to explain how the plot tells you that derivative

$$D[f[x[t], y[t]], t]$$

is positive at the higher point and negative at the lower point.

□G.1.c.ii)

Now add to the plot above the tangent vector and gradient vector at the point $\{x[\pi], y[\pi]\}$.

Again, remembering that

$$D[f[x[t], y[t]], t] = \text{gradf}[x[t], y[t]] \cdot \{x'[t], y'[t]\},$$

use what you see to explain why the derivative $D[f[x[t], y[t]], t]$ is equal to 0 at $t = \pi$.

Confirm what you say with a calculation of the derivative at that point.

□G.1.d.i)

You can describe any line through the point $\{1, 1\}$ by selecting a unit vector $\{a, b\}$ and putting:

$$\{x[t], y[t]\} = \{1, 1\} + t \{a, b\}.$$

Go with

$$f[x, y] = x^2 y$$

and say how you should set a and b so that the derivative of $f[x[t], y[t]]$ with respect to t is as big as possible when $t = 0$.

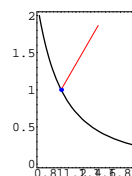
□G.1.d.ii)

Here is a true scale plot of the level curve

$$x^2 y = 1$$

together with a line through $\{1, 1\}$:

```
{a, b} = {1/2, \sqrt{3}/2};
Clear[x, y, t]
level = ContourPlot[x^2 y, {x, 0, 2}, {y, 0, 2}, Contours -> {1},
ContourShading -> False, DisplayFunction -> Identity];
line = ParametricPlot[{1, 1} + t {a, b}, {t, 0, 1},
PlotStyle -> {{Red, Thickness[0.01]}}, DisplayFunction -> Identity];
point = Graphics[{Blue, PointSize[0.04], Point[{1, 1}]}];
Show[level, line, point, AspectRatio -> Automatic,
DisplayFunction -> $DisplayFunction];
```



Copy, paste, and edit the code above so that it plots the line you found in the last part.

Describe what you see and explain why you see it.

G.2) The gradient is perpendicular to the level curves and surfaces*

□G.2.a)

Plot part of the level curve

$$f[x, y] = y - \sin[x] = 0$$

and throw in a few normal vectors.

□G.2.b)

The sphere of radius 2 centered at $\{1, 2, 0\}$ is the level surface of the function

$$f[x, y, z] = (x - 1)^2 + (y - 2)^2 + z^2 = 4.$$

It's not easy to use $f[x, y, z]$ to plot this sphere, but plotting it parametrically is not bad if you use:

```
Clear[s, t, sphereplotter]
sphereplotter[s_, t_] =
{1, 2, 0} + 2 {Sin[s] Cos[t], Sin[s] Sin[t], Cos[s]};
```

Check:

```
Clear[f, x, y, z]
f[x_, y_, z_] = (x - 1)^2 + (y - 2)^2 + z^2;
TrigExpand[f@@sphereplotter[s, t]]
```

4

Good.

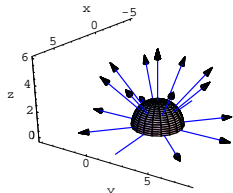
Here comes the plot of the top cap of the sphere with a selection of

gradient vectors

$$\nabla f[x, y, z] = \text{gradf}[x, y, z]$$

at some selected points on the cap:

```
Clear[gradf, gradvect]
gradf[x_, y_, z_] =
{D[f[x, y, z], x], D[f[x, y, z], y], D[f[x, y, z], z]};
topcap = ParametricPlot3D[Evaluate[sphereplotter[s, t]],
{s, 0, Pi/2}, {t, 0, 2 Pi}, DisplayFunction -> Identity];
gradvect[s_, t_] :=
Arrow[gradf@sphereplotter[s, t], Tail -> sphereplotter[s, t]];
Show[topcap,
Table[
gradvect[s, t], {t, 0, 7 Pi/4, Pi/4}, {s, 0, Pi/2, Pi/4}], ViewPoint -> CMView,
AxesLabel -> {"x", "y", "z"}, Boxed -> False, BoxRatios -> Automatic,
PlotRange -> All, DisplayFunction -> $DisplayFunction];
```



Describe the relationship between the gradient vectors and the surface and explain why you could have predicted the outcome in advance.

□G.2.c.i)

You are looking for the highest and lowest possible z coordinates of points {x, y, z} on the level surface f[x, y, z] = 1 for a function f[x, y, z].

Explain:

At {x₁, y₁, z₁} on the level surface with the highest z or the lowest z coordinate, the first two slots of gradf[x₁, y₁, z₁] must be 0.

□G.2.c.ii)

The points {x, y, z} on the sphere x² + y² + z² = 1 with the highest and lowest z coordinates are {0, 0, 1} and {0, 0, -1}.

Here's how you use the information in part i) to make Mathematica spit these points out:

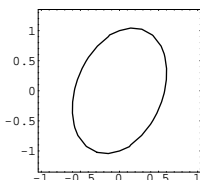
```
Clear[f, x, y, z]
f[x_, y_, z_] = x^2 + y^2 + z^2;
gradf[x_, y_, z_] =
{D[f[x, y, z], x], D[f[x, y, z], y], D[f[x, y, z], z]};
eqn1 = f[x, y, z] == 1;
eqn2 = D[f[x, y, z], x] == 0;
eqn3 = D[f[x, y, z], y] == 0;
Solve[{eqn1, eqn2, eqn3}]
{{z -> -1, x -> 0, y -> 0}, {z -> 1, x -> 0, y -> 0}}
```

The level curve

$$f[x, y] = 3x^2 - xy + y^2 = 1$$

is an ellipse in 2D:

```
Clear[f, x, y]
f[x_, y_] = 3x^2 - xy + y^2;
level = ContourPlot[f[x, y], {x, -1, 1}, {y, -1.3, 1.3},
Contours -> {1}, ContourSmoothing -> True, ContourShading -> False];
```



Adapt the idea immediately above to nail down the points {x, y} on this ellipse with the highest and lowest y coordinates.

G.3) The heat seeker*

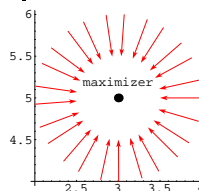
□G.3.a.i)

When you go with f[x, y] = 4 - ((x - 3)² + (y - 5)²), you can spot the maximizer at {3, 5}.

```
Clear[f, gradf, x, y]
f[x_, y_] = 4 - ((x - 3)^2 + (y - 5)^2);
gradf[x_, y_] = {D[f[x, y], x], D[f[x, y], y]};
N[Solve[gradf[x, y] == {0, 0}, {x, y}]]
{{x -> 3., y -> 5.}}
```

Look at this plot of gradf[x, y] at various points on the circle of radius 1 centered at the maximizer at {3, 5}:

```
maximizer = {3, 5};
maxpoint = {Graphics[{PointSize[0.05], Point[maximizer]}],
Graphics[Text["maximizer", maximizer, {0, -2}]]};
radius = 1;
Clear[x, y, t]
{x[t_], y[t_]} = maximizer + radius {Cos[t], Sin[t]};
scalefactor = 0.25;
gradients = Table[Arrow[gradf[x[t], y[t]], Tail -> {x[t], y[t]},
VectorColor -> Red, ScaleFactor -> scalefactor], {t, 0, 12, 0.5}];
Show[maxpoint, gradients, Axes -> True];
```



Why do you think that those scaled gradient vectors are pointing the way they are?

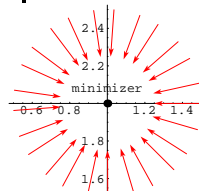
□G.3.a.ii)

When you go with f[x, y] = (x - 1)² + (y - 2)², you can spot the minimizer at {1, 2}.

```
Clear[f, gradf, x, y]
f[x_, y_] = (x - 1)^2 + (y - 2)^2;
gradf[x_, y_] = {D[f[x, y], x], D[f[x, y], y]};
N[Solve[gradf[x, y] == {0, 0}, {x, y}]]
{{x -> 1., y -> 2.}}
```

Look at this plot of -gradf[x, y] at various points on the circle of radius 0.5 centered at the minimizer, {1, 2}:

```
minimizer = {1, 2};
minpoint = {Graphics[{PointSize[0.04], Point[minimizer]}],
Graphics[Text["minimizer", minimizer, {0, -2}]]};
radius = 0.5;
Clear[x, y, t]
{x[t_], y[t_]} = minimizer + radius {Cos[t], Sin[t]};
scalefactor = 0.5;
negativegradients =
Table[Arrow[scalefactor (-gradf[x[t], y[t]]), Tail -> {x[t], y[t]},
VectorColor -> Red, ScaleFactor -> scalefactor], {t, 0, 12, 0.5}];
Show[minpoint, negativegradients, Axes -> True];
```



Why do you think that those scaled negative gradient vectors are pointing the way they are?

□G.3.b.i) The heat seeker

The Calculus&Mathematica Missile Company is working on some primitive heat seeking devices and you are chief engineer of the TAD (Target Acquisition Division). The current problem under study is to program a device to go to the hottest point in a temperature distribution.

For instance, if

$$\text{temp}[x, y] = \frac{100}{1 + (x - 2.5)^2 + 2(y - 3.5)^2}$$

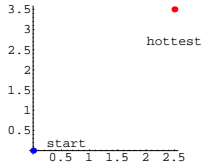
measures the temperature at a point {x, y}, then the hottest point is {2.5, 3.5} because the denominator is smallest at this point.

You can use the gradient to try to make a heat seeking device that starts at {0, 0} and tries to seek out the hottest spot {2.5, 3.5}.

Here's a look:

```
Clear[temp, gradtemp, x, y]
temp[x_, y_] = 100 / (1 + (x - 2.5)^2 + 2 (y - 3.5)^2);
```

```
gradtemp[x_, y_] = {D[temp[x, y], x], D[temp[x, y], y]};
hottestpoint = {2.5, 3.5};
start = {0, 0};
hotpt = Graphics[{{PointSize[0.04], Red, Point[hottestpoint]},
  Text["hottest", hottestpoint, {0, 4}]}];
startpt = Graphics[{{PointSize[0.04], Blue, Point[start]},
  Text["start", start, {-1.5, -1}]}];
setup = Show[hotpt, startpt, PlotRange -> All, Axes -> True,
  AxesOrigin -> {0, 0}, AspectRatio -> 1];
```



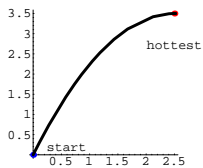
The heat seeker can't tell the exact location of the hot spot, but it can sense the gradient of the temp[x, y] merely by noting the hottest direction at a point {x, y}.
 If the heat seeker is at a point {x, y}, why should you program the heat seeker so that it leaves {x, y} in the direction of gradtemp[x, y]?

□G.3.b.ii)

Given:
 → The heat seeker can update its direction every instant.
 → The heat seeker is programmed so that it leaves {x, y} in the direction of gradtemp[x, y].
 Explain why the following plot displays a good approximation of the heat seeker's actual path when the seeker starts at {0, 0}:

```
Clear[Derivative, x, y, t]
equationx = x'[t] == gradtemp[x[t], y[t]][[1]];
equationy = y'[t] == gradtemp[x[t], y[t]][[2]];
starterx = x[0] == 0;
startery = y[0] == 0;
endtime = 10;
approxsolutions = NDSolve[{equationx, equationy, starterx, startery},
  {x[t], y[t]}, {t, 0, endtime}];
Clear[seeker]
seeker[t_] = {x[t] /. approxsolutions[[1]], y[t] /. approxsolutions[[1]]};
```

```
seekerplot = ParametricPlot[seeker[t], {t, 0, endtime},
  PlotStyle -> {{Thickness[0.02]}}, DisplayFunction -> Identity];
Show[setup, seekerplot, PlotRange -> All,
  DisplayFunction -> $DisplayFunction];
```



□G.3.b.iii)

Given:
 → The heat seeker can update its direction every instant.
 → The heat seeker is programmed so that it leaves {x, y} in the direction of gradtemp[x, y].
 Give a plot of the heat seeker's path when the seeker starts at {0, 2}.

□G.3.c.i) Bad news

The people over at the assembly division tell you that the heat seeker can't be built so as to update its direction at every instant. Instead, it will update its direction many times, but it will move on straight line segments between direction updates. Your group at TAD reacts to this information by programming the heat seeker as follows:
 → If the heat seeker is at {x_{k-1}, y_{k-1}}, then the heat seeker moves to a new point

$$\{x_k, y_k\} = \{x_{k-1}, y_{k-1}\} + \text{jump gradtemp}[x_{k-1}, y_{k-1}]$$

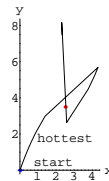
where the jump is positive number selected by trial and error.

For appropriately small jump numbers why is this a good update?

□G.3.c.ii)

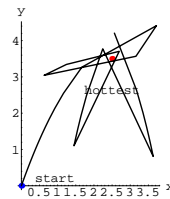
Now go from theory to practice.
 Start at {0, 0} and program the heat seeker with jump = 0.1 and 20 updates:

```
start = {0, 0};
hotpt = Graphics[{{PointSize[0.04], Red, Point[hottestpoint]},
  Text["hottest", hottestpoint, {0, 4}]}];
startpt = Graphics[{{PointSize[0.04], Blue, Point[start]},
  Text["start", start, {-1.5, -1}]}];
jump = 0.1;
updates = 20;
Clear[next, point, k]
next[{x_, y_}] = {x, y} + jump gradtemp[x, y];
point[0] = start;
point[k_] := point[k] = N[next[point[k - 1]]];
path = Graphics[{Thickness[0.01], Line[Table[point[k], {k, 0, updates}]]}];
Show[hotpt, startpt, path, Axes -> Automatic, AxesLabel -> {"x", "y"},
  AspectRatio -> Automatic];
```



Good start, but the heat seeker lost its cool. It nearly ran right over the hottest spot without even stopping to say hello.
 Reprogram it with a smaller jump and more updates:

```
jump = 0.04;
updates = 40;
Clear[next, point, k]
next[{x_, y_}] = {x, y} + jump gradtemp[x, y];
point[0] = start;
point[k_] := point[k] = N[next[point[k - 1]]];
path = Graphics[{Thickness[0.01], Line[Table[point[k], {k, 0, updates}]]}];
Show[hotpt, startpt, path, Axes -> Automatic, AxesLabel -> {"x", "y"},
  AspectRatio -> Automatic];
```



This time the heat seeker got close, but it blew its cool just as it was about to accomplish its desires.

Use trial and error to program in a jump size and an update number that send the heat seeker steadily to the hot spot so it can ignite its warhead and blow that hot spot to smithereens.

□G.3.c.iii)

One way to increase the efficiency of the heat seeker is to use one of the larger jump sizes at first and run it until it goes bats. Then use the last good point generated as a new starting point with a new, reduced jump size and run again.

Try this out starting at {0, 0} on the same function as above, incorporating any additional ideas that come to you.

□G.3.d)

Use jumps and updates, as above, to guide the heat seeker starting at {0, 0} to the hottest point for:

```
Clear[temp, gradtemp, x, y]
temp[x_, y_] =  $\frac{100}{1 + (2x - y)^2 + (x - 1)^2}$ ;
```

You can see that the hottest point is {1, 2}
 because $x = 1$ and $y = 2$
 make the denominator of $\text{temp}[x, y]$ as small as it can be.

G.4) Doing 'em by hand*

Ability to do these problems is a good sign of gradient literacy. You should be able to do them by hand over coffee with a pencil on a napkin or the back of an envelope.

G.4.a)

Lots of folks like to use the notations

$$\nabla f[x, y] = \text{gradf}[x, y]$$

and

$$\nabla f[x, y, z] = \text{gradf}[x, y, z].$$

Calculate the gradient $\nabla f[x, y]$ by hand for

$$f[x, y] = e^{2x-3y}$$

Calculate the gradient $\nabla f[x, y, z]$ by hand for

$$f[x, y, z] = \text{Cos}\left[\frac{xy}{z}\right].$$

G.4.b.i)

Lots of folks like to use the notations

$$f^{(1,0)}[x, y] = \frac{\partial f[x, y]}{\partial x} = D[f[x, y], x]$$

and

$$f^{(0,1)}[x, y] = \frac{\partial f[x, y]}{\partial y} = D[f[x, y], y]$$

Calculate

$$\frac{\partial f[x, y]}{\partial x} = f^{(1,0)}[x, y]$$

by hand for

$$f[x, y] = x^2 + y^2 + 1.$$

G.4.b.ii)

Calculate

$$\frac{\partial f[x, y]}{\partial y} = f^{(0,1)}[x, y]$$

by hand for

$$f[x, y] = e^{x-4y}.$$

G.4.b.iii)

Calculate

$$\frac{\partial f[x, y, z]}{\partial z} = f^{(0,0,1)}[x, y, z]$$

by hand for

$$f[x, y, z] = \text{Sin}[x^2 y^3 z^4].$$

G.4.c)

Here is

$$\frac{\partial f[x, y]}{\partial x} = f^{(1,0)}[x, y]$$

for

$$f[x, y] = e^{\text{Sin}[xy-x]}.$$

```
Clear[f, x, y]
f[x_, y_] = E^Sin[x y - x];
D[f[x, y], x]
-E^-Sin[x - x y] (1 - y) Cos[x - x y]
```

And here is

$$\frac{\partial f[1.8, 0.7]}{\partial x} = f^{(1,0)}[1.8, 0.7]$$

for the same function:

```
D[f[x, y], x] /. {x -> 1.8, y -> 0.7}
-0.153877
```

What does this calculation tell you about what happens to $f[x, y]$ when you hold y at 0.7 but make x just a teensy-weensy bit bigger than 1.8?

G.4.d)

Use the total differential of

$$f[x, y, z] = x y z$$

to give a hand derivation of the formula

$$D[x[t] y[t] z[t], t] = x'[t] y[t] z[t] + x[t] y'[t] z[t] + x[t] y[t] z'[t]$$

G.4.e)

Explain why the following notation gives you nothing more than the total differential of f . Think about the relationship between the gradient and the total differential.

$$\nabla f[x, y] \cdot \{dx, dy\} = \left\{ \frac{\partial f[x, y]}{\partial x}, \frac{\partial f[x, y]}{\partial y} \right\} \cdot \{dx, dy\}.$$

G.4.f) Derivative analogies

Comment on the analogy among the chain rule formulas:

$$\rightarrow D[f[x[t]], t] = f'[x[t]] x'[t].$$

$$\rightarrow D[f[x[t], y[t]], t] = \text{gradf}[x[t], y[t]] \cdot \{x'[t], y'[t]\}.$$

$$\rightarrow D[f[x[t], y[t], z[t]], t] = \text{gradf}[x[t], y[t], z[t]] \cdot \{x'[t], y'[t], z'[t]\}.$$

G.4.g)

The fundamental formula of calculus and the chain rule for functions of one variable give you the clean formula

$$\int_a^b g'[x[t]] x'[t] dt = g[x[b]] - g[x[a]].$$

Use the fundamental formula of calculus and the chain rule for functions of two variables to give a clean formula for

$$\int_a^b \text{gradf}[x[t], y[t]] \cdot \{x'[t], y'[t]\} dt.$$

Heavy Tip:

How does

$$\text{gradf}[x[t], y[t]] \cdot \{x'[t], y'[t]\}$$

relate to the derivative of

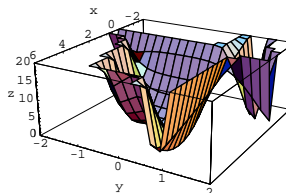
$$g[t] = f[x[t], y[t]]?$$

G.5) The highest crests and the deepest dips

G.5.a)

Way back in 1958, a fellow named Beale came up with a function whose plot looks like an Alpine valley. Here is a fairly careful rendition of the plot of his function:

```
Clear[f, x, y]
f[x_, y_] = (1.500 - x (1 - y^2))^2 + (2.250 - x (1 - y^3))^2 + (2.625 - x (1 - y^4))^2;
Plot3D[f[x, y], {x, -3, 6}, {y, -2, 2}, PlotPoints -> {20, 20},
PlotRange -> {0, 20}, ClipFill -> None, ViewPoint -> CMView,
AxesLabel -> {"x", "y", "z"}];
```



Look at:

```
Clear[gradf]
gradf[x_, y_] = {D[f[x, y], x], D[f[x, y], y]};
candidates = Chop[NSolve[gradf[x, y] == {0, 0}]]
{{x -> 0, y -> -0.240932 + 1.30684 I},
{x -> 0, y -> -0.240932 - 1.30684 I}, {x -> 0, y -> -1.37528},
{x -> 0, y -> 1.}, {x -> 0.0962699 - 0.0274921 I, y -> -1.26456 - 1.76695 I},
{x -> 0.0962699 + 0.0274921 I, y -> -1.26456 + 1.76695 I},
{x -> 1.97765 - 0.37012 I, y -> -0.518203 + 0.826085 I},
{x -> 1.97765 + 0.37012 I, y -> -0.518203 - 0.826085 I}, {x -> 2.125, y -> 0},
{x -> 2.24351, y -> -0.44166}, {x -> 5.31639, y -> 0.84052}}
```

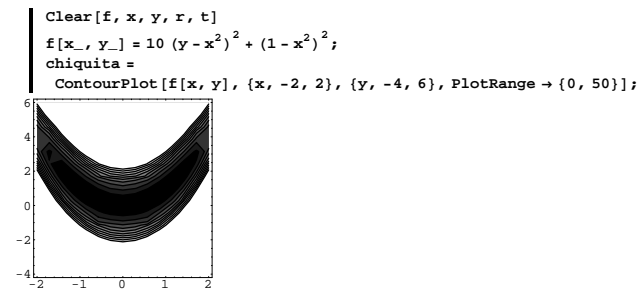
Use what you see above to locate the deepest point in this valley.

□G.5.b) The banana function

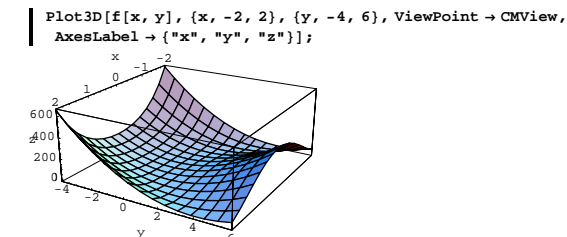
Lots of folks like to call the function

$$f[x, y] = 10(y - x^2)^2 + (1 - x^2)^2$$

"Rosenbrock's banana", because its level curves look like bananas and because it was first studied by someone named Rosenbrock. Take a look:



Here's a 3D plot of the banana function:



Note the parabolic valley running through the surface. Use `gradf[x, y]` to find the lowest points (= deepest dips) in this valley.

After you've got the locations, plant flagpoles at these locations so everyone can see them.

□G.5.d.i)

Analyze the formula

$$f[x, y] = \frac{\sin[2x] + \sin[3y]}{0.5 + x^2 + y^2}$$

and say how you know that the surface

$$z = f[x, y]$$

definitely has a deepest dip and a highest crest.

□G.5.d.ii)

Use contour plots, surface plots and the `FindMinimum` instruction to help come up with your own best estimates of the exact locations of the highest crest and the deepest dip on the surface you studied in part i) immediately above.

Once you've located them, plot the surface with flagpoles planted at the top of the highest crest and at the bottom of the deepest dip.

G.6) Closest points, gradients and Lagrange's method

□G.6.a.i)

Here's the level curve

$$f[x, y] = \left(\frac{x}{3}\right)^2 + y^2 = 1$$

shown with the point {2, 1.5}:

```
{a, b} = {2, 1.5};
Clear[f, x, y]
f[x_, y_] = (x/3)^2 + y^2;
levelcurve = ContourPlot[f[x, y], {x, 0, 3}, {y, 0, 1.7}, Contours -> {1},
ContourShading -> False, DisplayFunction -> Identity];
point = Graphics[{Red, PointSize[0.08], Point[{a, b}]}];
setup = Show[levelcurve, point, AspectRatio -> Automatic, Axes -> True,
AxesLabel -> {"x", "y"}, DisplayFunction -> $DisplayFunction];
```

□G.5.c)

The function studied in this problem was adapted from the book *Process Optimization with Applications in Metallurgy and Chemical Engineering* by W. H. Ray and J. Szekely, Wiley-Interscience, New York, 1973.

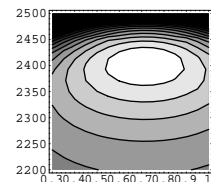
Engineers at the C&M Engineering and Foundry Company are analyzing the profit per hour from a solid state reaction carried on at a high temperature *t*. In their work, they come across a function like this:

```
Clear[f, x, t]
f[x_, t_] = 1/10^18 (10 (1 - 0.7 x) x t^3 E^0.01 t - E^0.02 t - 5 x (1 - x) - 0.1 t^4);
```

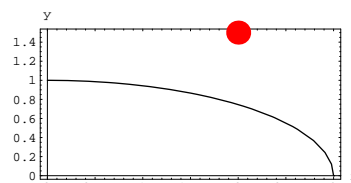
You are working in the math division of C&M and in comes some e-mail from the engineers asking for the *x* and *t* that make this function as large as possible. The engineers mention that *x* is a variable the engineers call "void fraction" and *t* is temperature at which the reaction is to run. This function takes you aback because there is no hope in setting its gradient equal to 0 and then solving for *x* and *t*.

You call down to the engineers and ask what reasonable ranges on the temperature *t* and *x* are. They reply that the reaction can run at temperatures between 2200° and 2500° and *x* must be between 0.3 and 1.0. You thank them and then you immediately look at:

```
ContourPlot[f[x, t], {x, 0.3, 1.0}, {t, 2200, 2500}];
```



You take one look at this and say, "Yippee!" because now you know how to use Mathematica to come up with the optimal *x* and *t*. Do it.



The first goal is to determine the point on the level curve that is closest to the given point {a, b} = {2, 1.5}.

The distance between the point {a, b} and a point {x, y} on the level curve is

$$\sqrt{(x - a, y - b) \cdot (x - a, y - b)}.$$

The square of the distance between the point {a, b} and a point {x, y} on the level is:

```
Clear[distsquared]
distsquared[x_, y_] = {x - a, y - b} \cdot {x - a, y - b}
(-2 + x)^2 + (-1.5 + y)^2
```

Lagrange's method is a natural to find the point on the level curve that is closest to {a, b}:

```
Clear[gradf, graddistsquared]
gradf[x_, y_] = {D[f[x, y], x], D[f[x, y], y]};
graddistsquared[x_, y_] =
{D[distsquared[x, y], x], D[distsquared[x, y], y]};
Clear[s]
candidates =
N[Solve[{f[x, y] == 1, gradf[x, y] == s graddistsquared[x, y]}]]
{{s -> -1.12712, x -> 1.82053, y -> 0.794821},
{s -> 0.0665134, x -> -2.98282, y -> -0.106879}, {s -> 0.284401 - 0.127684 I,
x -> 2.83114 + 0.612401 I, y -> -0.531471 + 0.362473 I},
{s -> 0.284401 + 0.127684 I, x -> 2.83114 - 0.612401 I,
y -> -0.531471 - 0.362473 I}}
```

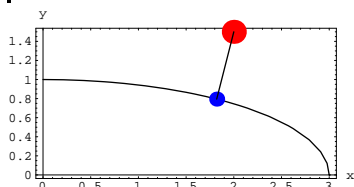
Test the viable candidates:

```
candidate1 = {1.82053, 0.794821};
candidate2 = {-2.98282, -0.106879};
{distsquared@candidate1, distsquared@candidate2}
{0.529487, 27.4106}
```

The closest point on the level curve $f[x, y] = 1$ to the given point {a, b} is candidate1.

Here is candidate1 shown, in true scale, with the line that connects the given point {a, b} to the closest point on the level curve::

```
closest = candidate1;
closestpoint = Graphics[{Blue, PointSize[0.05], Point[closest]};
connector = Graphics[Line[{{a, b}, closest}]];
outcome = Show[setup, closestpoint, connector];
```

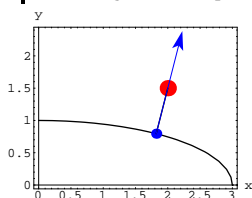


The connecting line seems to be perpendicular to the level curve. Is this an accident? Why or why not?

□G.6.a.ii)

Here is the plot immediately above shown with the gradient of f[x, y] calculated at the closest point:

```
Show[outcome, Arrow[gradf@closest, Tail -> closest],
PlotRange -> All, AspectRatio -> Automatic];
```

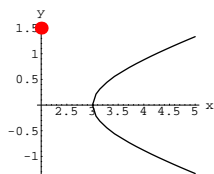


Bingo. Why did it have to happen this way?

□G.6.b.i)

Here's the level curve
 $f[x, y] = (\frac{x}{3})^2 - y^2 = 1$
 shown with the point {2, 1.5}:

```
{a, b} = {2, 1.5};
Clear[f, x, y]
f[x_, y_] = (x/3)^2 - y^2;
levelcurve = ContourPlot[f[x, y], {x, 3, 5}, {y, -1.5, 1.5},
Contours -> {1}, ContourShading -> False, DisplayFunction -> Identity];
point = Graphics[{Red, PointSize[0.08], Point[{a, b}]}];
setup = Show[point, levelcurve, AspectRatio -> Automatic, Axes -> True,
AxesLabel -> {"x", "y"}, DisplayFunction -> $DisplayFunction];
```



Use Lagrange's method to find the point on the level curve that is closest to the given point {a, b} = {2, 1.5} and plot, as above, the line connecting the given point to the closest point, and plot the gradient of f[x, y] calculated at the closest point with its tail at the closest point. Say why you are not totally surprised by what the plot displays.

□G.6.b.ii)

One of the GiveItA Try problems in the previous lesson brought to light the fact that if X and Y are 3D vectors, then $\|X \times Y\| = \|X\| \|Y\| |\sin[\text{angle between}]|$. Even if you didn't do that particular problem, it's a pretty good idea to know about this because it tells you that saying that two 3D vectors X and Y point in same or opposite directions is the same as saying that $X \times Y = \{0, 0, 0\}$.

Now take the same function and the same point {a, b} as in part i) above:

```
{a, b} = {2, 1.5};
Clear[f, x, y]
f[x_, y_] = (x/3)^2 - y^2
```

$$\frac{x^2}{9} - y^2$$

Put:

```
threeDgradf[x_, y_] = {D[f[x, y], x], D[f[x, y], y], 0}
{2x/9, -2y, 0}
```

And look at:

```
crossproduct = Cross[threeDgradf[x, y], {(x, y, 0) - {a, b, 0}}];
N[Solve[{f[x, y] == 1, crossproduct == {0, 0, 0}}]]
{{x -> -3.0132, y -> 0.0939042},
{x -> 1.71916 - 0.30432 I, y -> -0.070143 + 0.828746 I},
{x -> 1.71916 + 0.30432 I, y -> -0.070143 - 0.828746 I},
{x -> 3.17487, y -> 0.346382}}
```

Explain why this gives you the same candidates you got using Lagrange's method in part i).

G.7) The Cobb-Douglas manufacturing model for industrial engineering

Manufacturing costs are usually split into capital costs (CEO stock options, golden parachutes, limos, private jets, other overhead, equipment, etc.) and labor costs (salaries and sweat). Manufacturers can manipulate these expenses in many ways. They can automate heavily, or they can cut capital costs by increasing the labor force. Economists Cobb and Douglas became famous because they noticed that if a manufacturing process uses x units of capital and y units of labor, then the output function of a manufacturing process is usually approximated by

$$f[x, y] = A x^{k/m} y^{(m-k)/m}$$

for some fixed constants A, k, and m that depend on the goods being manufactured.

```
Clear[f, x, y, A, k, m, t]
f[x_, y_] = A x^{k/m} y^{m-k/m}
A x^{k/m} y^{m-k/m}
```

This makes some sense because if you look at:

```
{f[t x, t y], t f[x, y]}
```

$$\{A (t x)^{k/m} (t y)^{(m-k)/m}, A t x^{k/m} y^{m-k/m}\}$$

Then a little mental algebra revolving around the fact that

$$t^{k/m} t^{(m-k)/m} = t^{k/m + (m-k)/m} = t^{m/m} = t$$

tells you that

$$f[t x, t y] = t f[x, y],$$

and this means that the output is directly proportional to the inputs of capital and labor.

In this problem, you are a big shot in The C&M Manufacturing Co., and are preparing a production run of a certain product. The bean counters over in the finance office say that for a production run, capital costs will be \$129.15 for each unit x of capital and \$95.74 for each unit y of labor.

So the overall cost in dollars of a production run is

$$129.15 x + 95.74 y.$$

The company has budgeted exactly \$50,000 on this production run.

This gives you the budget line

$$129.15 x + 95.74 y = 50000.$$

The industrial engineers have figured out that that x units of labor and y units of capital result in a production run of

$$f[x, y] = 149.2 x^{3/4} y^{1/4} \text{ units.}$$

```
Clear[f, gradf, x, y, t]
f[x_, y_] = 149.2 x^{3/4} y^{1/4}
149.2 x^{3/4} y^{1/4}
gradf[x_, y_] = {D[f[x, y], x], D[f[x, y], y]}
{111.9 y^{1/4} / x^{1/4}, 37.3 x^{3/4} / y^{3/4}}
```

Here is a plot of the budget line

$$129.15 x + 95.74 y = 50000.$$

To get a useful plot of the line, pick off the two extreme possibilities:

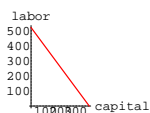
```
nolabor = Solve[129.15 x + 95.74 y == 50000 /. x -> 0]
{{y -> 522.248}}
point1 = {0, 522.25}
{0, 522.25}
alllabor = Solve[129.15 x + 95.74 y == 50000 /. y -> 0]
{{x -> 387.147}}
point2 = {387.15, 0}
```

```
{387.15, 0}
| {x[t_], y[t_]} = point1 + t (point2 - point1)
{387.15 t, 522.25 - 522.25 t}
```

As t runs from 0 to 1, the points {x[t], y[t]} sweep out the budget line from one extreme to the other.

The extreme at the left uses only labor and the extreme at the right uses only capital.

```
budgetline = ParametricPlot[{x[t], y[t]}, {t, 0, 1},
PlotStyle -> {{Red, Thickness[0.02]}}, AspectRatio -> Automatic,
AxesLabel -> {capital, labor}];
```



The points {x[t], y[t]} on this line represent all possible expenditures {capital, labor} that conform to the budget of \$50,000.

Find the t* between 0 and 1 such that production

$$f[x[t], y[t]]$$

is maximized at t = t*, and report the optimal values

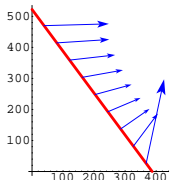
$$\{x[t^*], y[t^*]\} = \{\text{capital units, labor units}\}$$

that give the most production for the given budget.

□G.7.a.i)

Here are some gradient vectors gradf[x[t], y[t]] with their tails at a choice of points {x[t], y[t]} on the line plotted above; the plot is in true scale:

```
grads = Table[Arrow[gradf[x[t], y[t]],
Tail -> {x[t], y[t]}, {t, 0.1, 0.95, 0.85/8}];
Show[budgetline, grads, AspectRatio -> Automatic, AxesLabel -> None];
```



Take the maximizing t* you got in in the last part and add the plot of 2 gradf[x[t*], y[t*]]

with tail at {x[t*], y[t*]} to the plot immediately above.

The scale factor 2 is tacked on so that you can easily distinguish this gradient vector from the others.

Describe what you see and then discuss the information conveyed by the plots of the other gradient vectors above.

□G.7.a.ii)

Use the chain rule formula

$$D[f[x[t], y[t]], t] = \text{gradf}[x[t], y[t]] \cdot \{x'[t], y'[t]\}$$

to explain why your plot of

$$\text{gradf}[x[t^*], y[t^*]] \text{ with tail at } \{x[t^*], y[t^*]\}$$

turned out the way it did.

G.8) Data Fit in two variables: Plucking a guitar string

The ends of a guitar string are anchored at 0 and π on the x-axis and the string is pulled to an initial position and then allowed to vibrate on its own.

Experience shows that for certain strings the function that measures the height of the string above the point x on the x-axis at time t after the pluck is well-approximated by a function of the form

$$p[x, t] = a_1 \text{Sin}[x] \text{Cos}[t] + a_2 \text{Sin}[2x] \text{Cos}[2t] + a_3 \text{Sin}[3x] \text{Cos}[3t] + \dots$$

where 0 ≤ x ≤ π, and t > 0 is time measured after the pluck.

You use as many terms as you need for good results.

Adjustment of units can make this work for any string.

When you go with

$$p[x, t] = a_1 \text{Sin}[x] \text{Cos}[t] + a_2 \text{Sin}[2x] \text{Cos}[2t] + a_3 \text{Sin}[3x] \text{Cos}[3t] + \dots$$

then you get

$$p[0, t] = a_1 \text{Sin}[0] \text{Cos}[t] + a_2 \text{Sin}[0] \text{Cos}[2t] + a_3 \text{Sin}[0] \text{Cos}[3t] + \dots = 0$$

because Sin[0] = 0, and you get

$$p[\pi, t] = a_1 \text{Sin}[\pi] \text{Cos}[t] + a_2 \text{Sin}[2\pi] \text{Cos}[2t] + a_3 \text{Sin}[3\pi] \text{Cos}[3t] + \dots = 0$$

because Sin[kπ] = 0 no matter what integer k you go with.

The physical interpretation of

$$p[0, t] = p[\pi, t] = 0$$

is that the wire is anchored to the x-axis at the points x = 0 and x = π.

□G.8.a.i)

For p[x, t] of the above form, what is the first time t after t = 0 that the string can be counted on to return to the shape it had at t = 0?

□Tip:

How long before Cos[t], Cos[2t], Cos[3t], Cos[4t], ...

begin to repeat themselves?

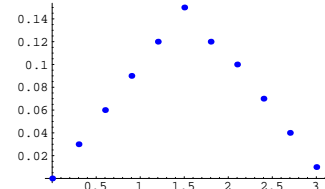
□G.8.a.ii)

As above the string, anchored at 0 and π on the x-axis, was plucked at t = 0 and the following data {x, p[x, 0]} measuring the height p[x, 0] of the string above position x on the x-axis at time t = 0 were collected:

```
pluckdata = {{0., 0.}, {0.3, 0.03}, {0.6, 0.06}, {0.9, 0.09},
{1.2, 0.12}, {1.5, 0.15}, {1.8, 0.12}, {2.1, 0.10}, {2.4, 0.07},
{2.7, 0.04}, {3., 0.01}, {N[π], 0.}};
```

Here is a plot(not in true scale).

```
pluckdataplot = ListPlot[pluckdata,
PlotStyle -> {PointSize[0.02], RGBColor[0, 0, 1]}];
```



Fit these data (for t = 0) as well as you can by a function

$$p[x, t] = a_1 \text{Sin}[x] \text{Cos}[t] + a_2 \text{Sin}[2x] \text{Cos}[2t] + a_3 \text{Sin}[3x] \text{Cos}[3t].$$

Show your estimated shape of the string at times t = 0, π/2, π, 3π/2, and 2π.

Plot your p[x, t] for 0 ≤ x ≤ π and 0 ≤ t ≤ 2π.

What do the cross sections cut by planes perpendicular to the t-axis represent?

□Tip:

When t = 0, all the cosine factors are equal to 1; so you want to fit the data with

$$a_1 \sin x + a_2 \sin 2x + a_3 \sin 3x.$$

After you get your values of a_1 , a_2 , and a_3 , then form your function

$$p[x, t] = a_1 \sin x \cos t + a_2 \sin 2x \cos 2t + a_3 \sin 3x \cos 3t.$$

□G.8.b)

The string in this part is not the same as the string in part a.ii) immediately above.

Cameras are set to monitor what a vibrating string anchored at 0 and π on the x-axis is doing at positions

$$x = 1.5 \text{ and } x = 2.5.$$

The cameras record the string's position at 0, 1, 2, 3, 4, and 5 seconds after it was plucked. Here is what the cameras found in the form $\{x, t, p[x, t]\}$ where x is position on the x-axis, t is time after the pluck and $p[x, t]$ is the height of the string above x at time t :

```
data = {{1.5, 0, 0.35}, {2.5, 0, 0.15}, {1.5, 1, 0.14}, {2.5, 1, 0.13},
        {1.5, 2, -0.10}, {2.5, 2, -0.11}, {1.5, 3, -0.34}, {2.5, 3, -0.15},
        {1.5, 4, -0.17}, {2.5, 4, -0.15}, {1.5, 5, 0.07}, {2.5, 5, 0.07}};
```

Get a good fit of these data by a function of the form

$$p[x, t] = a_1 \sin x \cos t + a_2 \sin 2x \cos 2t + a_3 \sin 3x \cos 3t + \dots$$

Then give a plot of $p[x, 0]$ to give your estimate of the shape of the initial pluck.

Finally make a movie showing how the string vibrates.

G.9) Linearizations and total differentials*

For functions $f[x]$ of one variable, lots of folks like to look at the linearization of $f[x]$ at point a :

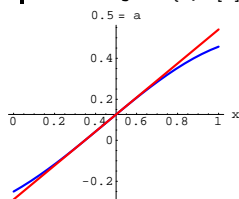
$$\text{linearf}[x, a] = f'[a](x - a) + f[a].$$

An example:

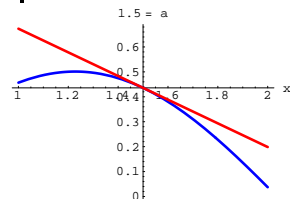
```
Clear[f, x, linf, a]
f[x_] = 1/3 Sin[1.7 x] - 1/4 Cos[2.1 x];
linearf[x_, a_] = f'[a](x - a) + f[a]
-1/4 Cos[2.1 a] + 1/3 Sin[1.7 a] +
(-a + x) (0.566667 Cos[1.7 a] + 0.525 Sin[2.1 a])
```

Here are plots of $f[x]$ and $\text{linearf}[x, a]$ for a couple of points a :

```
a = 0.5;
linplot1 = Plot[{f[x], linearf[x, a]}, {x, a - 0.5, a + 0.5},
PlotStyle -> {{Blue, Thickness[0.01]}, {Red, Thickness[0.01]}},
AxesOrigin -> {a, f[a]}, AxesLabel -> {"x", ""}, PlotLabel -> a = "a";
```



```
a = 1.5;
linplot2 = Plot[{f[x], linearf[x, a]}, {x, a - 0.5, a + 0.5},
PlotStyle -> {{Blue, Thickness[0.01]}, {Red, Thickness[0.01]}},
AxesOrigin -> {a, f[a]}, AxesLabel -> {"x", ""}, PlotLabel -> a = "a";
```



The linearized version of $f[x]$ at a point a is just the tangent line through $\{a, f[a]\}$, but it has some strong virtues:

- As x varies, $\text{linearf}[x, a]$ is much easier to calculate than $f[x]$.
 - $\text{linearf}[x, a]$ approximates $f[x]$ very well for x 's near a .
 - $\text{linearf}[x, a]$ goes up if $f[x]$ goes up as x advances through a .
 - $\text{linearf}[x, a]$ goes down if $f[x]$ goes down as x advances through a .
- In short, if x is close to a , then $\text{linearf}[x, a]$ mimics $f[x]$ very well.

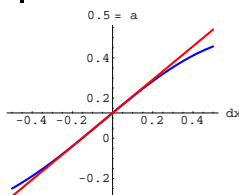
Lots of other folks (especially older folks) prefer to look at the differential of $f[x]$ at a point a :

$$df = f'[a] dx$$

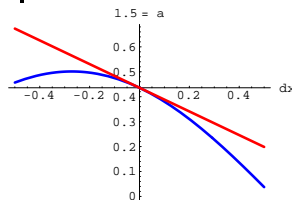
They treat dx as a number and say $f[a + dx]$ is well approximated by $f[a] + df$.

See what this means for the same function at the same points looked at above.

```
Clear[df, dx]
a = 0.5;
df = f'[a] dx;
diffplot1 = Plot[{f[a + dx], f[a] + df}, {dx, -0.5, 0.5},
PlotStyle -> {{Blue, Thickness[0.01]}, {Red, Thickness[0.01]}},
AxesOrigin -> {0, f[a]}, AxesLabel -> {"dx", ""}, PlotLabel -> a = "a";
```



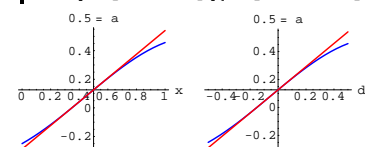
```
Clear[df, dx]
a = 1.5;
df = f'[a] dx;
diffplot2 = Plot[{f[a + dx], f[a] + df}, {dx, -0.5, 0.5},
PlotStyle -> {{Blue, Thickness[0.01]}, {Red, Thickness[0.01]}},
AxesOrigin -> {0, f[a]}, AxesLabel -> {"dx", ""}, PlotLabel -> a = "a";
```



□G.9.a)

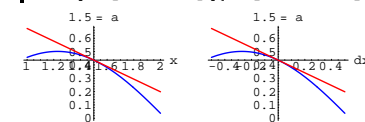
Here is linplot1 compared to diffplot1:

```
Show[GraphicsArray[{linplot1, diffplot1}]];
```



Here is linplot2 compared to diffplot2:

```
Show[GraphicsArray[{linplot2, diffplot2}]];
```



Rerun for other functions and other points to get enough experience to answer the question: Is the first approach (linearizations) doing different things from the second approach (differentials), or are the two approaches doing the same thing but describing it differently?

□G.9.b.i)

For functions of two or more variables, the same two approaches often persist.

For functions $f[x, y]$ of two variables, lots of folks like to look at the linearization of $f[x, y]$ at point $\{a, b\}$:

$$\text{linearf}[x, y, a, b] = \text{gradf}[a, b] \cdot \{x - a, y - b\} + f[a, b].$$

An example:

```
Clear[f, x, y, gradf, linearf, a, b]
f[x_, y_] = 5 - 1/2 (x^2 + y^2);
gradf[x_, y_] = {D[f[x, y], x], D[f[x, y], y]}
{-x, -y}
linearf[x_, y_, a_, b_] = gradf[a, b] . {x - a, y - b} + f[a, b]
5 + 1/2 (-a^2 - b^2) - a(-a + x) - b(-b + y)
```

Here is a plot showing both $f[x, y]$ and $\text{linearf}[x, y, a, b]$ for $\{a, b\} = \{-1, 1\}$.

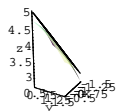
```
{a, b} = {-1, 1};
h = 0.5;
```

```

plot = ParametricPlot3D[{x, y, f[x, y]},
  {x, a - h, a + h}, {y, b - h, b + h}, DisplayFunction -> Identity];
fplot = Insert[plot, EdgeForm[], {1, 1}];
linfplot =
  ParametricPlot3D[{x, y, linearf[x, y, a, b]}, {x, a - h, a + h},
  {y, b - h, b + h}, PlotPoints -> {7, 7}, DisplayFunction -> Identity];
linearfplot = linfplot /. Polygon -> Line;

both = Show[fplot, linearfplot, ViewPoint -> CMView,
  PlotRange -> All, AxesLabel -> {"x", "y", "z"}, Boxed -> False,
  DisplayFunction -> $DisplayFunction];

```

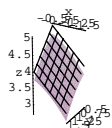


See it from a different viewpoint:

```

newlook =
  Show[both, ViewPoint -> {-4, 7, 4}, AxesLabel -> {"x", "y", "z"}];

```



The two plots touch at {a, b}:

```

{f[x, y], linearf[x, y, a, b]} /. {x -> a, y -> b}
{4, 4}

```

The curved surface is $f[x, y]$ and the plane plastered to it at the point $\{a, b, f[a, b]\}$ is the plot of $\text{linearf}[x, y, a, b]$. Assess the quality of the approximation of $f[x, y]$ by $\text{linearf}[x, y, a, b]$ as x and y vary in the vicinity of $\{a, b\} = \{-1, 1\}$.

Tip:

The plot tells you a lot, but if you are happier with cold numbers, look at values of $f[x, y]$ and $\text{linearf}[x, y]$ for some points $\{x, y\}$ near the point of linearization at $\{a, b\} = \{-1, 1\}$:

```

d = 0.2;
PaddedForm[
  TableForm[Flatten[Prepend[Table[{x, y, f[x, y], linearf[x, y, a, b]},
    {x, a - d, a + d, d}, {y, b - d, b + d, d}],
    {"x", "y", "f value", "linear f value"}], 1],
  TableAlignments -> Center],
  {3, 2}]

```

x	y	f value	linear f value
-1.20	0.80	3.96	4.00
-1.20	1.00	3.78	3.80
-1.20	1.20	3.56	3.60
-1.00	0.80	4.18	4.20
-1.00	1.00	4.00	4.00
-1.00	1.20	3.78	3.80
-0.80	0.80	4.36	4.40
-0.80	1.00	4.18	4.20
-0.80	1.20	3.96	4.00

□G.9.b.ii)

The folks who prefer differentials get in on the same act. They take the total differential of a function $f[x, y]$ at a point $\{a, b\}$

$$df = f^{(1,0)}[a, b] dx + f^{(0,1)}[a, b] dy = \text{gradf}[a, b] \cdot \{dx, dy\}$$

They treat dx and dy as numbers and say

$$f[a + dx, b + dy]$$

is well approximated by

$$f[a, b] + df.$$

See what this means for the same function $f[x, y]$ at the same point you looked at above:

```

Clear[f, x, y, gradf, a, b]
f[x_, y_] = 5 - 1/2 (x^2 + y^2);
gradf[x_, y_] = {D[f[x, y], x], D[f[x, y], y]}
{-x, -y}

```

Here is a plot showing both

$$f[a + dx, b + dy] \text{ and } f[a, b] + df$$

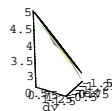
for $\{a, b\} = \{-1, 1\}$.

```

Clear[dx, dy, df]
{a, b} = {-1, 1};
df = gradf[a, b] . {dx, dy}
dx - dy
h = 0.5;
fplotd = ParametricPlot3D[{dx, dy, f[a + dx, b + dy]},
  {dx, -h, h}, {dy, -h, h}, DisplayFunction -> Identity];
dfplotd =
  ParametricPlot3D[{a + dx, b + dy, f[a, b] + df}, {dx, -h, h}, {dy, -h, h},
  PlotPoints -> {7, 7}, DisplayFunction -> Identity] /. Polygon -> Line;

bothd =
  Show[fplotd, dfplotd, ViewPoint -> CMView, AxesLabel -> {"dx", "dy", ""},
  PlotRange -> All, Boxed -> False, DisplayFunction -> $DisplayFunction];

```

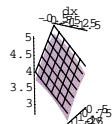


See it from a different viewpoint:

```

newlookd = Show[bothd, ViewPoint -> {-4, 7, 4}];

```



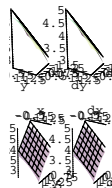
Look familiar?

See all four plots:

```

Show[GraphicsArray[{{both, bothd}, {newlook, newlookd}}]];

```



For functions of two variables, is the first approach (linearizations) doing different things from the second approach (differentials), or are the two approaches doing the same thing but describing it differently? Which approach do you prefer?

□G.9.c)

As you may have guessed, the guy who wrote this problem prefers the idea of linearizations to the idea of differentials, but sometimes differentials are just the ticket for making quick estimates.

Take:

```

Clear[f, gradf, x, y]
f[x_, y_] = 10.4 (1/3 (x + 2y) - (xy^2)^(1/3));
gradf[x_, y_] = {D[f[x, y], x], D[f[x, y], y]}
{10.4 (1/3 - y^2/(3(xy^2)^(2/3))), 10.4 (2/3 - 2xy/(3(xy^2)^(2/3)))}

```

You are sitting at $\{a, b\} = \{0.5, 1\}$ and you have two choices:

→ Increase x from a to $a + dx$ for a small number dx .

→ Increase y from b to $b + dy$ for a small number dy .

To see which of these choices should you take to increase $f[x, y]$ as much as you can, look at:

```

Clear[dx, dy, df]
{a, b} = {0.5, 1};
df = N[gradf[a, b] . {dx, dy}]
2.28108 dx - 4.9225 dy

```

Increasing x a little bit will drive $f[x, y]$ up.

Increasing y a little bit will drive $f[x, y]$ down.

Check:

```
dx = 0.2;
dy = 0.2;
{f[a+dx, b], f[a, b], f[a, b+dy]}
{1.13055, 0.648292, 0.114164}
```

Yep. To drive $f[x, y]$ up, increase x and forget about y .

Handy little tool for making quick estimates.

You go with

$$f[x, y] = 128.3 + 608.7 \left((x^2 y^3)^{1/5} - \frac{1}{5} (2x + 3y) \right).$$

You are sitting at $\{a, b\} = \{4.7, 2.9\}$ and you have two choices:

→ Increase x from a to $a + dx$ for a small number dx .

→ Increase y from b to $b + dy$ for a small number dy .

Which should you do to drive $f[x, y]$ up?

Which should you do to drive $f[x, y]$ down?

□Tip:

You're getting tired; so as a gesture of good will and friendship, here is some code:

```
Clear[f, gradf, x, y]
f[x_, y_] = 128.3 + 608.7 \left( -\frac{1}{5} (2x + 3y) + (x^2 y^3)^{1/5} \right);
gradf[x_, y_] = {D[f[x, y], x], D[f[x, y], y]}
{608.7 \left( -\frac{2}{5} + \frac{2xy^3}{5(x^2 y^3)^{4/5}} \right), 608.7 \left( -\frac{3}{5} + \frac{3x^2 y^2}{5(x^2 y^3)^{4/5}} \right)}
Clear[dx, dy, df]
{a, b} = {4.7, 2.9};
df = N[gradf[a, b] . {dx, dy}]
-61.2401 dx + 77.8114 dy
```

G.10 Keeping track of constituent costs

Calculus&Mathematica first learned of this problem from Dick Duffin of Carnegie Mellon University, Pittsburgh, Pennsylvania.

The location is near Pittsburgh on the bank of the Monongehela River not far from the old Westinghouse East Pittsburgh plant.

You've got the job of shipping V cubic feet of old steel shavings across the river by barge. To do this, you'll make a box to put the shavings in for the barge shipment.

Put

x = length of the box in feet,
 y = width of the box in feet, and
 z = height of the box in feet.

The material for the box consists of five rectangular pieces.

The bottom has area $x y$; the material for it costs $\$a$ per square foot. So in dollars,

$$\text{bottom cost} = a x y.$$

Another two pieces (the sides) have area $x z$ each; these cost $\$b$ per square foot. So in dollars

$$\text{side cost} = 2 b x z.$$

And the other two pieces (the ends) have area $y z$; these cost $\$c$ per square foot. So in dollars,

$$\text{end cost} = 2 c y z.$$

The tugboat captain tells us that she will charge d dollars for each trip.

You have V cubic feet of shavings; so you'll need $\frac{V}{xyz}$ trips. So in dollars,

$$\text{transportation cost} = \frac{dV}{xyz}.$$

The bottom line is

overall cost = bottom cost + side cost + end cost + transportation cost.

What should the ratios

$$\text{bottom cost} : \text{side cost} : \text{end cost} : \text{transportation cost}$$

be if the overall cost is at a minimum?

Give a formula for the minimum cost and the corresponding optimal dimensions, x , y , and z , in terms of a , b , c , d , and V .

How does the minimum cost change if the captain changes her mind and charges $2d$ dollars per trip?

G.11 The great pretender

□G.11.a.i)

If you are dealing with a function $f[x]$ of one variable and you find a crest at $\{x_0, f[x_0]\}$ and you find that $f'[x] = 0$ only at $x = x_0$, then can you be sure that x_0 maximizes $f[x]$?

□G.11.a.ii)

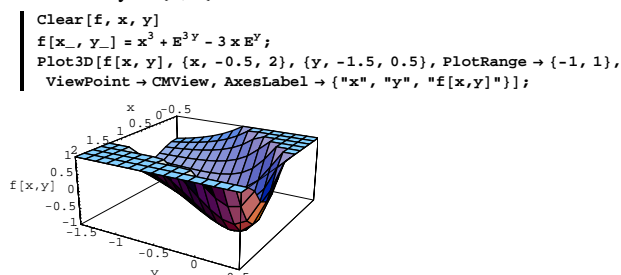
If you are dealing with a function $f[x]$ of one variable and you find a dip at $\{x_0, f[x_0]\}$ and you find that $f'[x] = 0$ only at $x = x_0$, then can you be sure that x_0 minimizes $f[x]$?

□G.11.b)

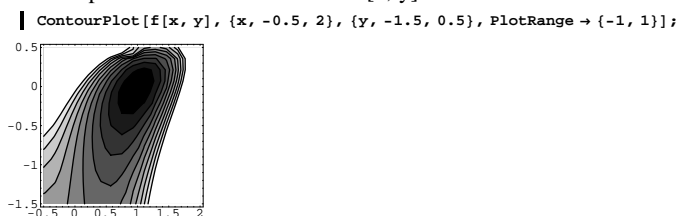
Here's a plot of

$$f[x, y] = x^3 + e^3 y - 3 x e^y$$

in the vicinity of $\{1, 0\}$.



Here is a plot of some level curves of $f[x, y]$.



The darker the shading, the lower the function.

□G.11.b.i)

See the dip?

Locate the bottom of the dip by finding where the gradient of $f[x, y]$ is $\{0, 0\}$.

```
Clear[gradf]
gradf[x_, y_] = {D[f[x, y], x], D[f[x, y], y]};
NSolve[gradf[x, y] == {0, 0}]
{{Log[1/x^2] -> -1. y, Log[1/x] -> -2. y}}
```

□G.11.b.ii)

Look at:

```
gradf[x, y]
{-3 E^y + 3 x^2, 3 E^3 y - 3 E^y x}
```

Are there any other $\{x_0, y_0\}$'s at which the gradient of

$$f[x, y] = x^3 + e^3 y - 3 x e^y$$

is $\{0, 0\}$?

□G.11.b.iii)

Is the point you found in part i) a genuine minimizer of

$$f[x, y] = x^3 + e^3 y - 3 x e^y$$

or just a pretender?

In fact, does

$$f[x, y] = x^3 + e^3 y - 3 x e^y$$

even have a minimizer?

□G.11.c)

True or false:

If you have a function $f[x, y]$ with the properties that

→ The plot of $f[x, y]$ has a dip at $\{x_0, y_0, f[x_0, y_0]\}$;

→ The only solution of

$$\text{gradf}[x, y] = \{0, 0\}$$

is $x = x_0$ and $y = y_0$,

then $\{x_0, y_0\}$ is the minimizer of $f[x, y]$.