How Can You Visualize Green's Theorem?

Note: You may notice differences between this Maple worksheet and the equivalent Mathematica notebook. These differences were introduced to preserve the content of these modules and were necessary because of major functional differences between Maple and Mathematica.

Introduction

OBJECTIVE: Visualize a force field with a path superimposed on it to enhance geometrical insight into Green's Theorem, and compute line integrals over vector fields using parametrizations.

If you can visualize a force field and a curve through it, how can that help you understand Green's Theorem? In this module, you will explore integration over vector fields and use parameterizations to compute line integrals. You will also explore how you can determine the closed curve around which your work integral is a maximum and whether or not it makes any difference if a force is conservative.

Technology Guidelines

NOTE: If you have just finished a worksheet, **restart** *Maple* before executing a new worksheet.
TO OPEN SECTIONS,
Click on the **PLUS** sign at the left hand side of the screen *or* select **Expand All Sections** from the **View** drop down menu.

TO STOP AN EXECUTION
Click on **STOP** button from the toolbar.

ORDER OF EXECUTION
Execute commands in the order given. Do not skip any *Maple* Input lines within a given worksheet

Alternatively, you can execute the entire worksheet by selecting the **Execute Worksheet** command from the **Edit** drop down menu.

SAVING WORKSHEETS.

You can save anytime to any directory you choose, and it is wise to save often.
EXPERIENCING MAJOR PROBLEMS
Save if appropriate, and then shut down *Maple* and start it up again.

Part I: Green's Theorem (Circulation-Curl Form)

Visualizing the Problem

Start by loading in a package that will allow you to display the vector field. Be certain to load that package before you attempt to plot a vector field, and recall that a package should

be loaded only once.

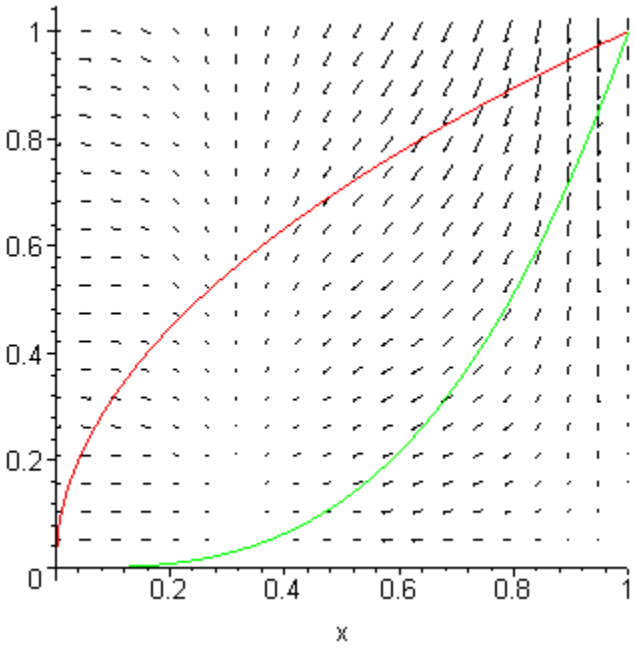
```
> restart:
  with(plots):
```

Warning, the name changecoords has been redefined

The force field is given as a vector with component *Cos*(*5x*) in the horizontal direction and (*-3x y*) in the vertical direction. We will plot two paths between (0, 0) and (1, 1) on this force field: $y = \sqrt{x}$ and $y = x^3$.

```
> force:=[cos(5*x),-3*x*y];
  pv:=fieldplot(force, x=0..1, y=0..1):
  pc:=plot({sqrt(x),x^3}, x=0..1):
  display({pv,pc});
```

$$force := [\cos(5x), -3xy]$$



If you travel in a counterclockwise direction (first along $y = x^3$ and then back along $y = \sqrt{x}$) from (0,0) to (1,1) and then back again, can you predict if the work done by the force will be positive or negative or 0? Recall that the integrand of the work integral is the dot product of the force and a vector in the direction of motion.

Computing the Line Integrals

Find the work done in traveling along the lower part of the closed curve x^3 . Note the following parametrization for that curve. In this section we use linear algebra functions from the **linalg** package. We load this package now.

```
> with(linalg):
```

Warning, the protected names norm and trace have been redefined and unprotected

```
> x:=t:
  y:=t^3:
  r1:={x,y}:
  dr1:=[diff(x,t),diff(y,t)]:
  w1:=evalf(int(dotprod(force,dr1),t=0..1)):
  print(`The first portion of work done is`, w1);
```

The first portion of work done is, -1.477499141

Does the fact that this answer is negative fit with what you might have predicted? Next, find the work done in traveling along the upper part of the closed curve $\sqrt{x}$. As before, you need an appropriate parametrization. Note the direction in which you were traveling. You are now at (1, 1) and need to return to (0, 0).

```
> unassign('x,y,t'):
  x:=t:
  y:=sqrt(t):
  r2:=[x,y]:
  dr2:=diff(r2,t):
  w2:=evalf(int(dotprod(force,dr2),t=1..0)):
  print(`The second portion of the work done is`, w2);
```

The second portion of the work done is, 0.9417848549

Now, add the two together to get the total work.

```
> workaroundclosedpath:=w1+w2:
  print(`The work done in travelling around the closed path is`, workaroundclosedpatl
```

The work done in travelling around the closed path is, -0.5357142861

Applying Green's Theorem

Apply Green's Theorem, and verify that your answers agree. We allow for a tolerance level of .001 in case numerical integration has to be used at any point. You **must** clear *x* and *y* before performing this next integration.

```
> unassign('x,y');
```

```
inside:=int(int(-diff(force[1],y)+diff(force[2],x), y=x^3..sqrt(x)),x=0..1.);
if (workaroundclosedpath<=inside and workaroundclosedpath>=inside-.001) then pr
(`Green's theorem is validated`) else print(`there's a problem`): fi;
```

$$inside := -0.5357142857$$

Green's theorem is validated

The integrand for the double integral in Green's Theorem is the curl of the force function dotted into a unit vector perpendicular to the element of
area; in this case, this is in the positive z direction. You can explore the curl vector function in the Java applet, "Concept of the Curl."

You Try It: Part I

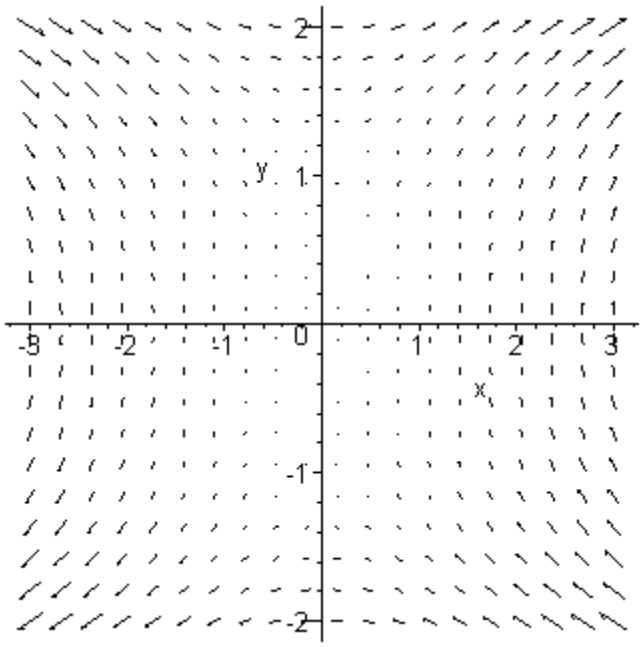
Is the Force Conservative?

Section 16.4 Exercise 34

In this problem, you are asked to find the closed path over which the work integral is a maximum. The force function given is $\left[\frac{x^2 y}{4}+\frac{y3}{3}, x\right]$

Begin by plotting the vector field.

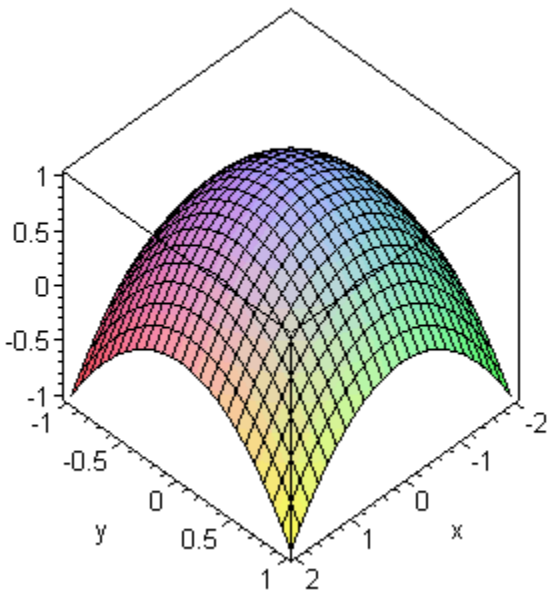
```
> unassign('x,y,force,pv,t');
  force:=[1/4*x^2*y+1/3*y^3, x]:
  pv:=fieldplot(force, x=-3..3, y=-2..2):
  display(pv);
```



Next, examine the integrand for the double integral that is equivalent, per Green's Theorem, to the work integral.

```
> circintegrand:=diff(force[2],x)-diff(force[1],y):
  print(`The integrand is`, circintegrand );
  plot3d(circintegrand, x=-2..2, y=-1..1, axes=boxed);
```

$$\textit{The integrand is,} \; -\frac{x^2}{4} - y^2 + 1$$



Note that the circulation integrand is nonnegative only when $\frac{x^2}{4}+y^2 \le 1$, that is, only

when you are inside the ellipse represented by the equality. Represent that ellipse parameterically by entering the correct functions for *x* and *y*.

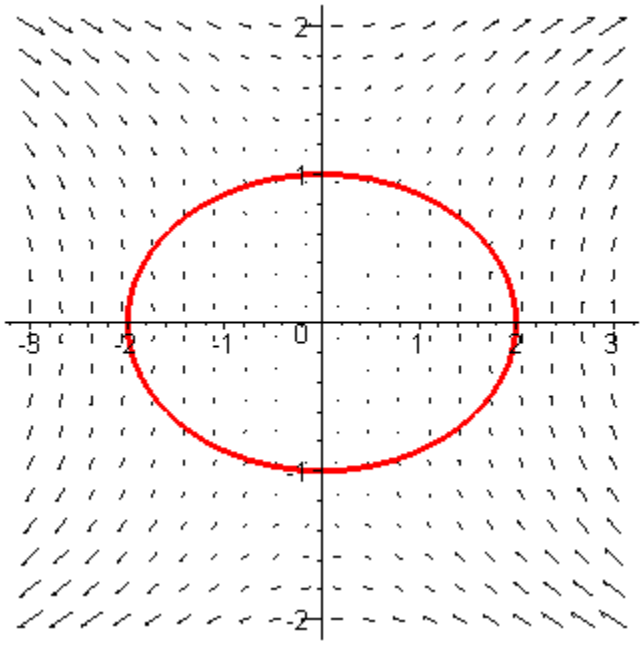
```
> x:=2*cos(t);
  y:=sin(t);
```

$$x := 2\cos(t)$$

$$y := \sin(t)$$

Consider a plot of your force field together with this ellipse. If your parametrization is correct and your bounds on *t* close the path, you should see your ellipse plotted with the force field by executing the next set of commands. Adjust the domain if necessary. If your plot looks wrong, go back and double check your parametrization.

```
> r:=[x,y]:
  pp:=plot([op(r), t=0..2*Pi], thickness=3):
  display(pp,pv);
```



Compute the work done in traveling around the ellipse. Adjust the domain if necessary.

```
> dr:=diff(r,t):
  Digits:=(7):
  workaroundclosedpath:=evalf(int(dotprod(force,dr),t=0..2*Pi));
```

$$workaroundclosedpath := 3.141593$$

Verify this using Green's Theorem.

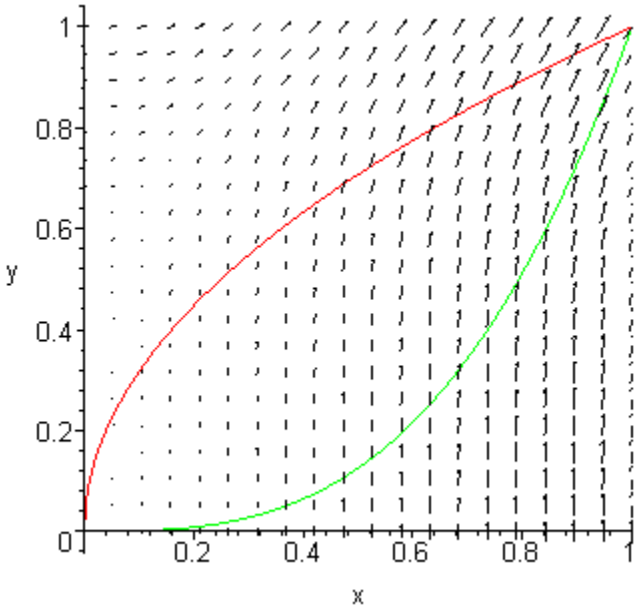
```
> unassign('x','y');
  inside:=int(int(circintegrand, y=-sqrt(1-x^2/4)..sqrt(1-x^2/4)),x=-2..2.0);
  if (workaroundclosedpath<=inside and workaroundclosedpath>=inside-.001) then pr
  (`Green's theorem is validated`) else print(`there's a problem`): fi;
```

$$inside := 3.141593$$

Green's theorem is validated

Just for comparison, determine the work done by this force in traveling around the curve in the first part of this lab. First look at the picture.

```
> unassign('x,y,t');
  pv2:=fieldplot(force, x=0..1,y=0..1):
  display(pv2,pc);
```



Now compute the work done in traveling around this closed path with the current force field.

```
> unassign('x','y','t');
  x:=t:
  y:=t^3:
  r1:=[x,y]:
  dr1:=diff(r1,t):
  w1:=int(dotprod(force,dr1),t=0..1.0):
  unassign('x','y','t'):
  x:=t:
  y:=sqrt(t):
```

```
r2:=[x,y]:
dr2:=diff(r2,t):
w2:=int(dotprod(force,dr2),t=1.0..0):
print(`The work around the closed path is`, evalf(w1+w2));
```

The work around the closed path is, 0.2869048

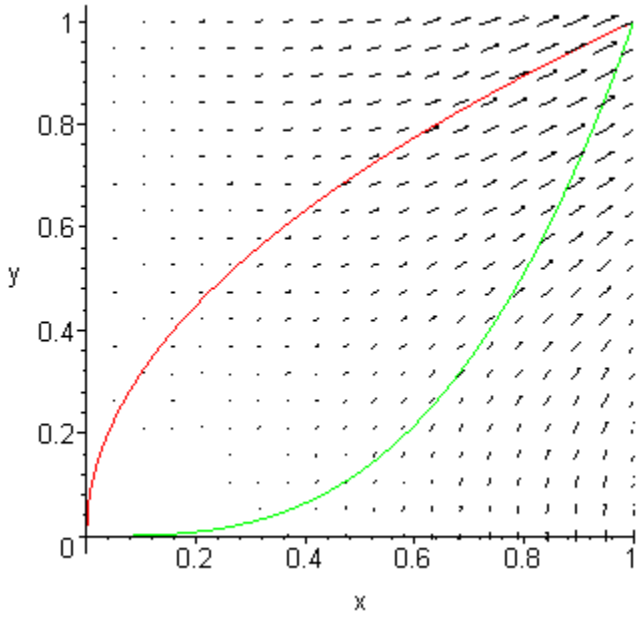
Is this answer smaller or larger than the work done in traveling around the ellipse?

Choose a Conservative Force and See what Happens

What if you had chosen a conservative force? You can check this out by merely going back to the force function and replacing it with a different function. Nothing else has to be changed. Following is an example, $[2xy, x^2]$ of a conservative force. Try out any others you want. Remember to leave spaces between variables when defining functions.

First, we will look at a plot.

```
> unassign('x,y,z'):
  force:=[2*x*y,x^2]:
  pv:=fieldplot(force, x=0..1, y=0..1):
  pc:=plot({sqrt(x),x^3}, x=0..1):
  display(pv,pc);
```



Next, we wil find the work done on each segment.

```
> unassign('x,y,t'):
```

```
x:=t:
y:=t^3:
r1:=[x,y]:
dr1:=diff(r1,t):
w1:=int(dotprod(force,dr1), t=0..1):
unassign('x,y,t'):
x:=t:
y:=sqrt(t):
r2:=[x,y]:
dr2:=diff(r2,t):
w2:=int(dotprod(force,dr2),t=1..0):
print(`The first portion of work done is`, w1); print(`The second portion of work don
```

The first portion of work done is, 1

The second portion of work done is, -1

As long as the force you use is conservative, how should the work done on the first portion be related to the work done on the second portion?

```
> print(`The work done in traveling around the closed path is`, w1+w2);
```

The work done in traveling around the closed path is, 0

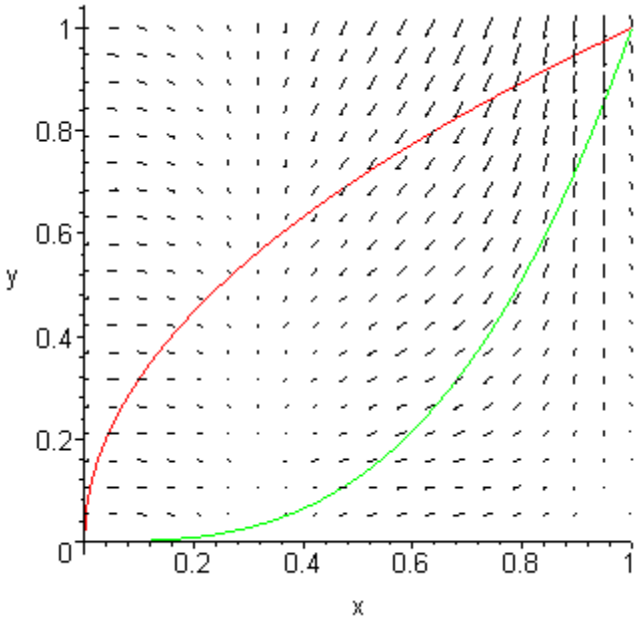
Part II: Green's Theorem (Flux-Divergence Form)

Visualizing the Problem

We use the **plots** package loaded in Part I to compute field plots.

As in Part I, the force field is given as a vector with component *Cos*(*5x*) in the horizontal direction and (*-3x y*) in the vertical direction. We will plot two paths between (0, 0) and (1, 1) on this force field: $\sqrt{x}$ and x^3

```
> unassign('x,y,z,force'):
  force:=[cos(5*x), -3*x*y]:
  pv:=fieldplot(force, x=0..1, y=0..1):
  pc:=plot({sqrt(x),x^3}, x=0..1):
  display(pv,pc);
```



If you travel in a counterclockwise direction (first along $y = x^3$ and then back along $y = \sqrt{x}$) from (0,0) to (1,1) and then back again, can you predict whether the outward flux will be positive or negative or 0? Recall that the integrand of the flux integral is the dot product of the force and a vector **perpendicular to** the direction of motion.

Computing the Line Integrals

Find the flux in traveling along the lower part of the closed curve. Choose an appropriate parameterization.

```
> x:=t:
  y:=t^3:
  r1:=[x,y]:
  dr1:=diff(r1,t):
  perp1:=[dr1[2],-dr1[1]]:
  flux1:=evalf(int(dotprod(force,perp1),t=0..1.)):
  print(`The flux over the first portion is`, flux1);
```

The flux over the first portion is, 0.1387527

Does the fact that this answer is small fit with what you might have predicted?

Now find the flux along the upper part of the closed curve. As before, choose an appropriate parametrization. Note the direction in which you are traveling. Continue traveling in the same direction as above.

```
> x:=t:
  y:=sqrt(t):
  r2:=[x,y]:
  dr2:=diff(r2,t):
  perp2:=[dr2[2],-dr2[1]]:
  flux2:=evalf(int(dotprod(force,perp2), t=1..0)):
  print(`The flux over the second portion is`, flux2);
```

The flux over the second portion is, -1.384100

Now, add the two together to get the total outward flux along the closed path.

```
> fluxclosedpath:=flux1+flux2:
  print(`The flux around the closed path is`, fluxclosedpath);
```

The flux around the closed path is, -1.245347

Applying Green's Theorem

Apply Green's Theorem, and verify that your answers agree. We allow for a tolerance level of .001 in case numerical integration has to be used at any point. You **must** clear *x* and *y* before performing this next integration.

```
> unassign('x','y'):
  inside:=int(int(diff(force[1],x)+diff(force[2],y), y=x^3..sqrt(x)),x=0..1.0);
  if (fluxclosedpath<=inside  and fluxclosedpath>=inside-.001) then print(`Green's theo
  validated`) else print(`there's a problem`): fi;
```

$$inside := -1.245347$$

Green's theorem is validated

You Try It: Part II

- **Choose a Conservative Force and See what Happens**

Do conservative forces mean anything when computing flux integrals? What type of force would give the flux around a closed curve to be 0? You can check these issues out by merely going back to the force function and replacing it with a different function - paste over the red with either one of the functions suggested or another one of your own. Nothing else has to be changed.

Suggested functions:

```
> force:=[2*x*y, x^2];
```

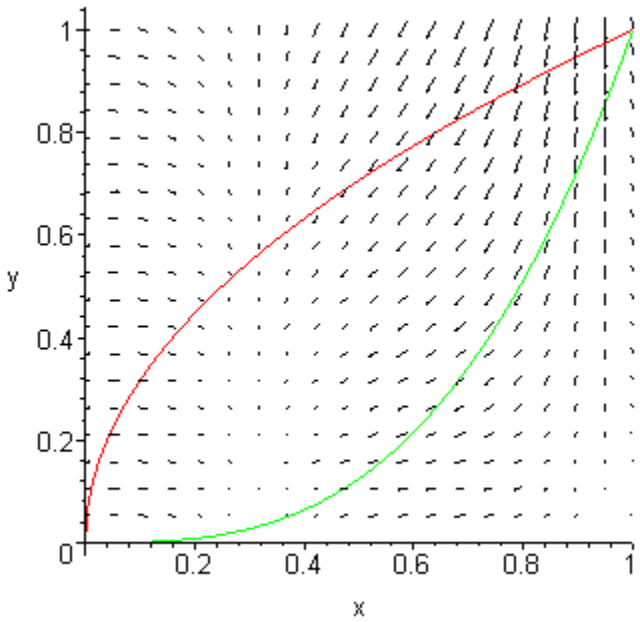
$$force := [2\,x\,y, x^2]$$

```
> force:=[x*sin(y),cos(y)];
```

$$force := [x \sin(y), \cos(y)]$$

The first set of commands defines and plots your functions.

```
> unassign('x,y,z,force'):
  force:=[cos(5*x), -3*x*y]:
  pv:=fieldplot(force,x=0..1,y=0..1):
  pc:=plot({sqrt(x),x^3}, x=0..1):
  display(pv,pc);
```



The next set of commands finds the associated flux integrals.

```
> unassign('x,y,t'):
  x:=t:
  y:=t^3:
  r1:=[x,y]:
  dr1:=diff(r1,t):
  perp1:=[dr1[2],-dr1[1]]:
  flux1:=evalf(int(dotprod(force,perp1),t=0..1)):
  unassign('x,y,t'):
  x:=t:
  y:=sqrt(t):
  r2:=[x,y]:
  dr2:=diff(r2,t):
  perp2:=[dr2[2],-dr2[1]]:
  flux2:=int(dotprod(force,perp2),t=1.0..0):
  print(`The flux over the first portion is`, flux1); print(`The flux over the second portion is`,
```

flux2);

The flux over the first portion is, 0.1387527

The flux over the second portion is, -1.384100

Put these results together.

> **fluxaroundclosedpath:=flux1+flux2:**
print(`The flux around the closed path is`,fluxaroundclosedpath);

The flux around the closed path is, -1.245347

Now verify Green's Theorem for this case.

> **unassign('x,y'):**
inside:=int(int(diff(force[1],x)+diff(force[2],y), y=x^3..sqrt(x)), x=0..1.);
if (fluxclosedpath<=inside and fluxclosedpath>=inside-.001) then print(`Green's theorem i validated`) else print(`there's a problem`): fi;

inside := -1.245347

Green's theorem is validated

>

Does the fact that the force field was conservative make any difference here?