

# Maple Tips

## Chapter 13

### Lesson 13-1

#### Defining a Vector Field

The **VectorField** command in the **VectorCalculus** package can be used to define vector fields. Note that you can use the **SetCoordinates** command to define the coordinate system for the entire worksheet so as to avoid the required '**cartesian**'[x,y,z] option.

```
> restart: with(VectorCalculus):
```

```
Warning, the assigned names <,> and <|> now have a global binding
```

```
Warning, these protected names have been redefined and unprotected: *, +, -, .., D, Vector, diff, int, limit, series
```

```
> fld1:=VectorField(<y,x>, 'cartesian'[x,y]);
```

$$fld1 := y \bar{e}_x + x \bar{e}_y$$

```
> SetCoordinates(cartesian[x,y,z]);
```

$$cartesian_{x,y,z}$$

```
> fld2:=VectorField(<y,z,x>);
```

$$fld2 := y \bar{e}_x + z \bar{e}_y + x \bar{e}_z$$

#### Graphing Vector Fields

The **fieldplot** and **fieldplot3d** commands in the **plots** package can be used to graph vector fields in 2D and 3D, respectively. The **grid=[a,b,c]** option can be used to change the number of vector rows in each dimension.

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

```
Warning, the name changecoords has been redefined
```

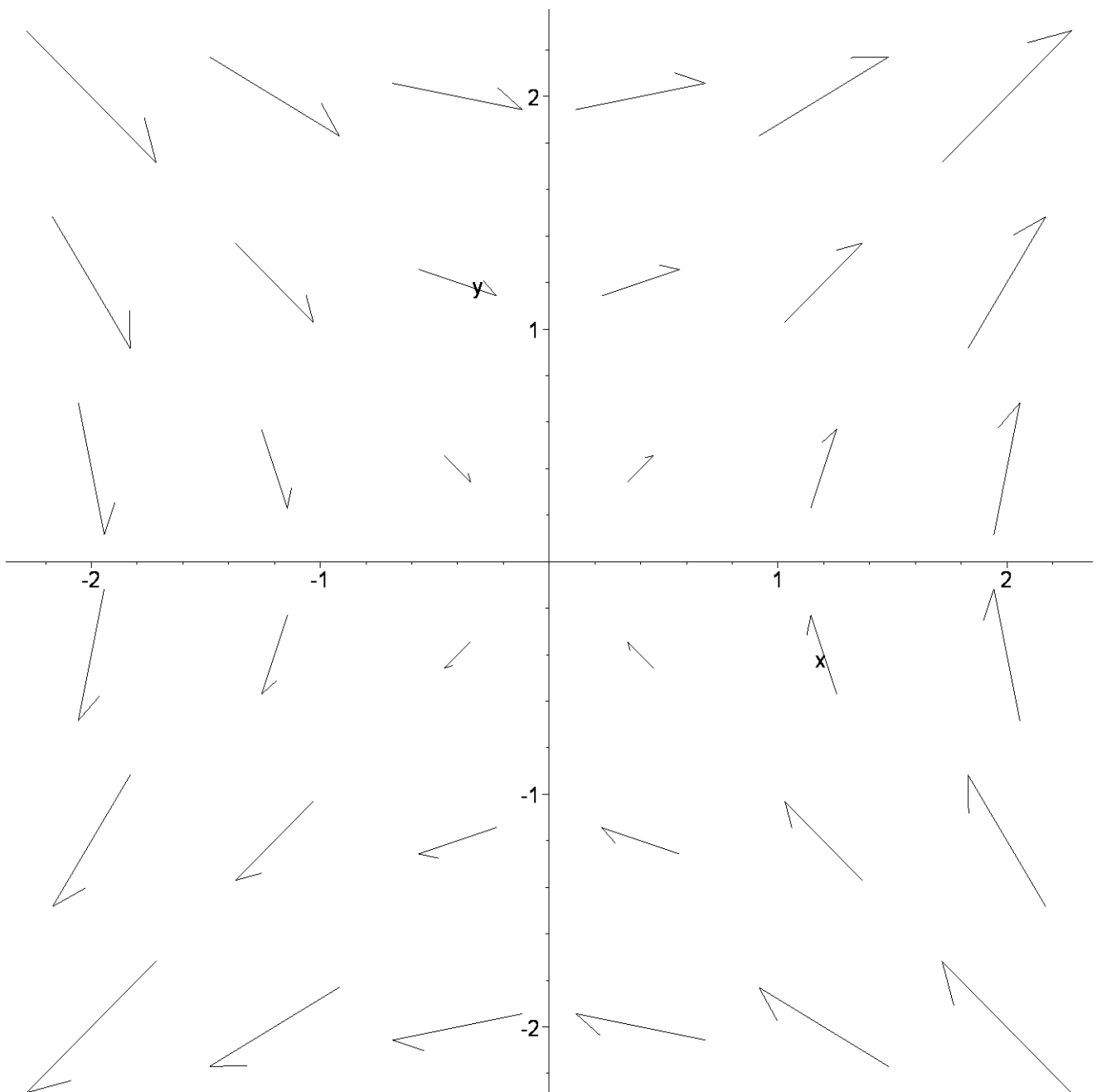
```
Warning, the assigned names <,> and <|> now have a global binding
```

```
Warning, these protected names have been redefined and unprotected: *, +, -, .., D, Vector, diff, int, limit, series
```

```
> fld1:=VectorField(<y,x>, 'cartesian'[x,y]);
```

$$fld1 := y \bar{e}_x + x \bar{e}_y$$

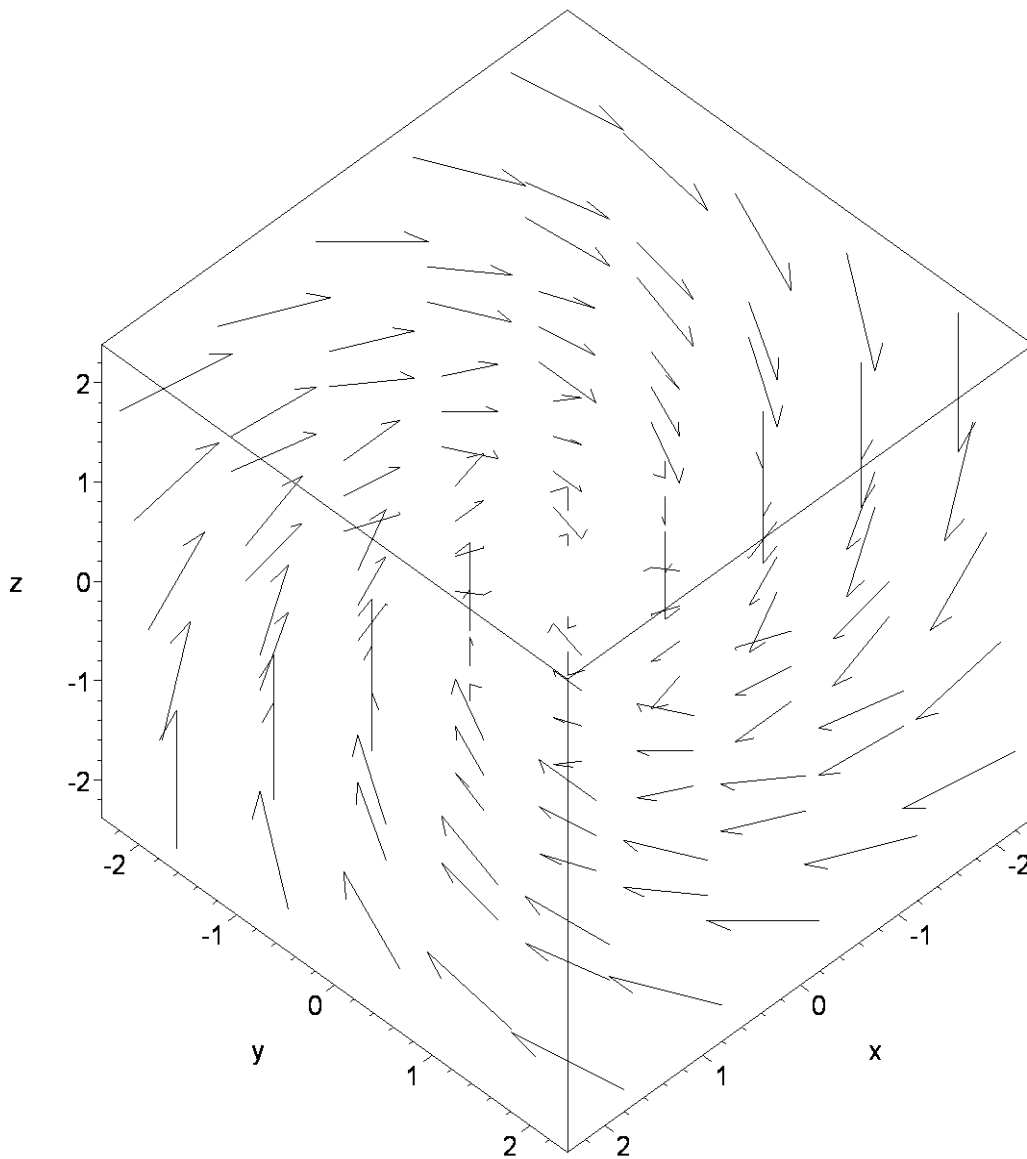
```
> fieldplot(fld1, x=-2..2, y=-2..2, grid=[6,6]);
```



```
> fld2:=VectorField(<y,z,x>, 'cartesian'[x,y,z]);
```

$$fld2 := y \bar{e}_x + z \bar{e}_y + x \bar{e}_z$$

```
> fieldplot3d(fld2, x=-2..2, y=-2..2, z=-2..2, grid=[5,5,5],
color=black, axes=box);
```



## Graphing Gradient Vector Fields

The `gradplot` and `gradplot3d` commands in the `plots` package can be used to graph gradient vector fields in 2D and 3D, respectively, given the potential function.

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

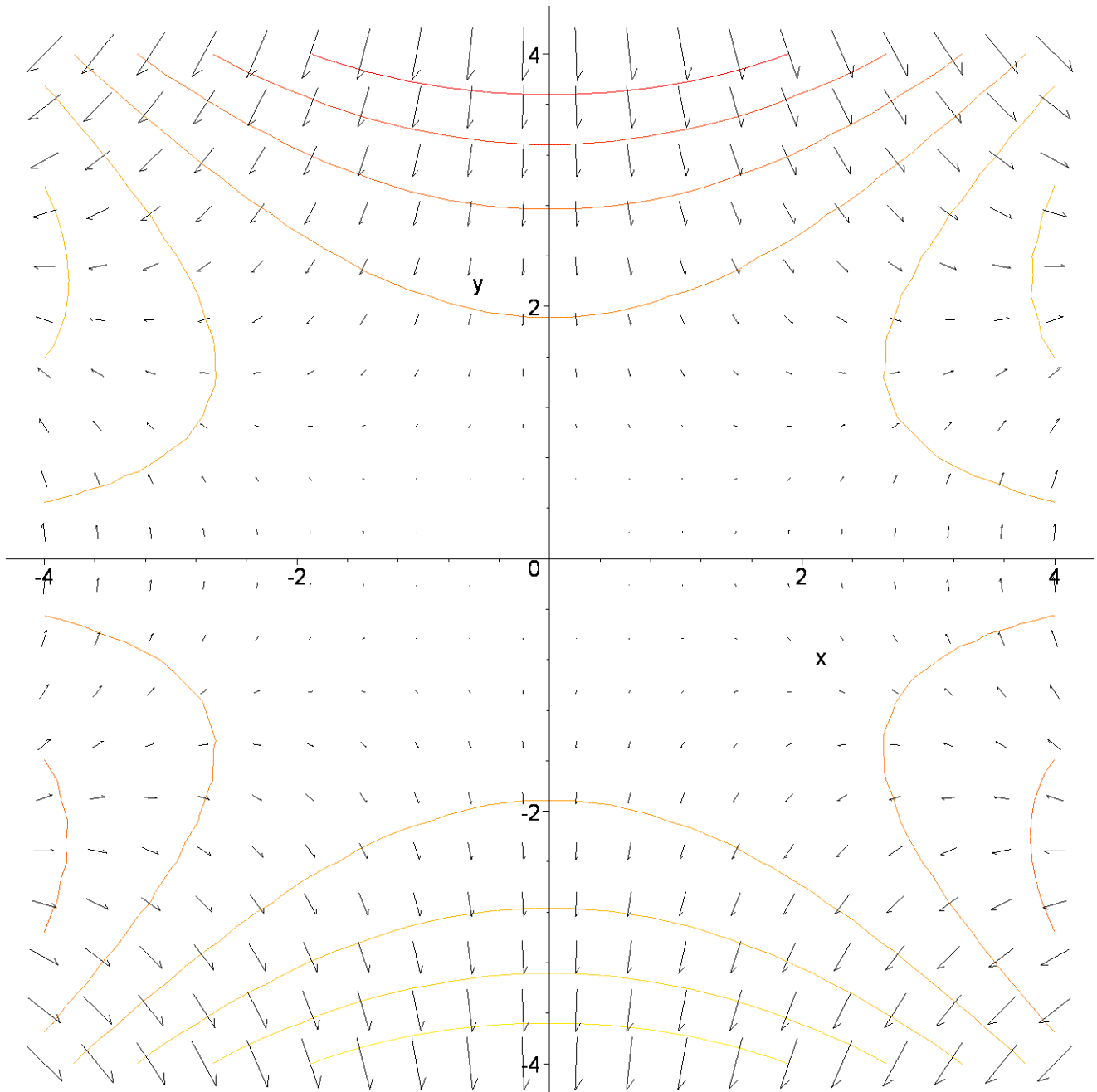
```
Warning, the name changecoords has been redefined
```

```
> f1:=(x,y)->x^2*y-y^3: f1(x,y);
```

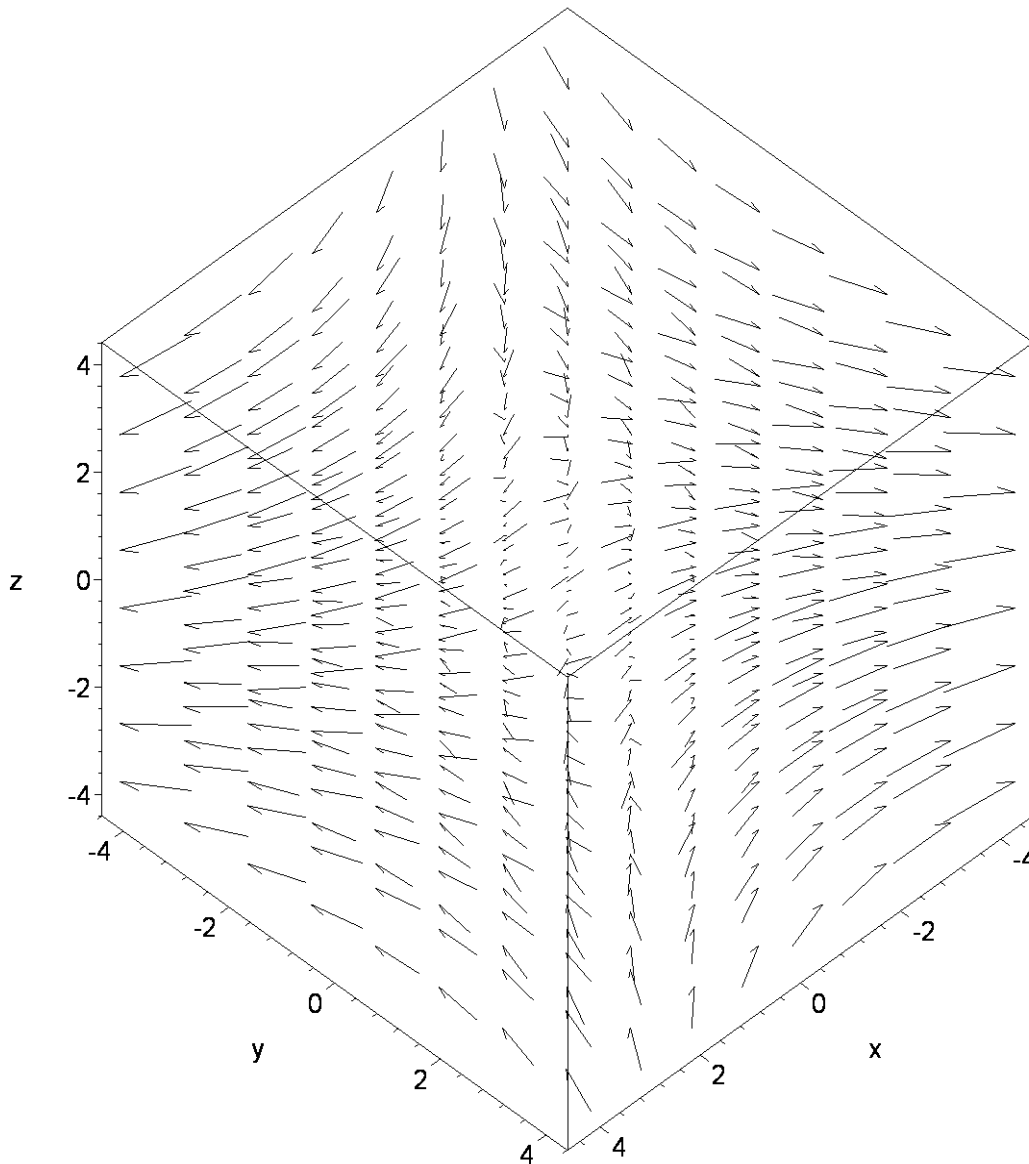
$$x^2y - y^3$$

```
> a:=gradplot(f1(x,y), x=-4..4, y=-4..4):
```

```
> b:=contourplot(f1(x,y), x=-4..4, y=-4..4):  
> display(a,b);
```



```
> f2:=(x,y,z)->x^2-2*x*y-z^2: f2(x,y,z);  
                                 $x^2 - 2xy - z^2$   
> gradplot3d(f2(x,y,z), x=-4..4, y=-4..4, z=-4..4, color=black,  
axes=box);
```



## Lesson 13-2

### Calculating a Line Integral in a Plane or Space

The **PathInt** command in the **VectorCalculus** package can be used to calculate line integrals in terms of arc length. There are several different options for defining the curve in the integral - **Line**, **LineLegments**, and **Path** are illustrated here.

```
> restart: with(VectorCalculus):
```

```
Warning, the assigned names <,> and <|> now have a global binding
```

Warning, these protected names have been redefined and unprotected: \*, +, -, ., D, Vector, diff, int, limit, series

```
> PathInt(2+x^2*y, [x,y]=Path(<cos(t), sin(t)>, t=0..Pi));
```

$$\frac{2}{3} + 2\pi$$

```
> PathInt(2*x, [x,y]=Line(<1,1>, <1,2>));
```

$$2$$

```
> PathInt(z+y*sin(x), [x,y,z]=LineSegments(<0,0,0>, <0,1,1>, <Pi,0,1>));
```

$$\frac{\sqrt{2}}{2} + \frac{\sqrt{1+\pi^2}(1+\pi)}{\pi}$$

## Calculating a Line Integral of a Vector Field

The **LineInt** command in the **VectorCalculus** package can be used to calculate line integrals of vector fields. Note the slight difference in defining the curves path versus the **PathInt** command - this is because the **VectorField** command already defines the coordinate system.

```
> restart: with(VectorCalculus):
```

Warning, the assigned names <, > and <|> now have a global binding

Warning, these protected names have been redefined and unprotected: \*, +, -, ., D, Vector, diff, int, limit, series

```
> LineInt(VectorField(<x^2, -x*y>, 'cartesian'[x,y]), Path(<cos(t), sin(t)>, t=0..Pi/2));
```

$$\frac{-2}{3}$$

```
> LineInt(VectorField(<y, z, x>, 'cartesian'[x,y,z]), LineSegments(<2,0,0>, <3,4,5>, <3,4,0>));
```

$$\frac{19}{2}$$

## Lesson 13-3

### Finding the Potential Function

The **ScalarPotential** command in the **VectorCalculus** package can be used to find the potential function to a conservative vector field. If the vector field is not conservative, the command will give no output at all (as in the last example below). Note that the constant of integration isn't shown in the output either.

```
> restart: with(VectorCalculus):
```

Warning, the assigned names <, > and <|> now have a global binding

Warning, these protected names have been redefined and unprotected: \*, +, -, ., D, Vector, diff, int, limit, series

```
> ScalarPotential(VectorField(<3+2*x*y, x^2-3*y^2>, 'cartesian'[x,y]));
```

$$3x + x^2y - y^3$$

```
> ScalarPotential(VectorField(<y^2, 2*x*y+exp(3*z), 3*y*exp(3*z)>, 'cartesian'[x,y,z]));
```

$$y^2x + ye^{(3z)}$$

```
> ScalarPotential(VectorField(<x-y, x-2>, 'cartesian'[x,y]));
```

## Lesson 13-4

### Verifying Green's Theorem

Use the **LineInt** command in the **VectorCalculus** package along with the **int** command to verify Green's Theorem.

```
> restart: with(VectorCalculus):
```

Warning, the assigned names <,> and <|> now have a global binding

Warning, these protected names have been redefined and unprotected: \*, +, -, ., D, Vector, diff, int, limit, series

```
> fld:=VectorField(<x^4, x*y>, 'cartesian'[x,y]);
```

$$fld := x^4 \bar{e}_x + xy \bar{e}_y$$

```
> LineInt(fld, LineSegments(<0,0>, <1,0>, <0,1>, <0,0>));
```

$$\frac{1}{6}$$

```
> int(int(diff(fld[2],x)-diff(fld[1],y), y=0..1-x), x=0..1);
```

$$\frac{1}{6}$$

## Lesson 13-5

### Calculating the Curl of a Vector Field

The **Curl** command in the **VectorCalculus** package can be used to calculate the curl of a vector field.

```
> restart: with(VectorCalculus):
```

Warning, the assigned names <,> and <|> now have a global binding

Warning, these protected names have been redefined and unprotected: \*, +, -, ., D, Vector, diff, int, limit, series

```
[ > fld:=VectorField(<x*z, x*y*z, -y^2>, 'cartesian'[x,y,z]):
> Curl(fld);
```

$$(-2y - xy)\bar{e}_x + x\bar{e}_y + yz\bar{e}_z$$

### Calculating the Divergence of a Vector Field

The **Divergence** command in the **VectorCalculus** package can be used to calculate the divergence of a vector field.

```
[ > restart: with(VectorCalculus):
Warning, the assigned names <,> and <|> now have a global binding

Warning, these protected names have been redefined and unprotected: *, +, -, ., D,
Vector, diff, int, limit, series

[ > fld:=VectorField(<x*z, x*y*z, -y^2>, 'cartesian'[x,y,z]):
> Divergence(fld);

z + xz

[ > Divergence(Curl(fld));
```

0

## Lesson 13-6

### Calculating a Surface Integral

The **SurfaceInt** command in the **VectorCalculus** package can be used to calculate surface integrals. There are several different options for defining the surface in the integral - **Surface**, **Sphere**, and **Box** are illustrated here.

```
[ > restart: with(VectorCalculus):
Warning, the assigned names <,> and <|> now have a global binding

Warning, these protected names have been redefined and unprotected: *, +, -, ., D,
Vector, diff, int, limit, series

[ > SurfaceInt(y, [x,y,z]=Surface(<u,v,u+v^2>, u=0..1, v=0..2));

13√2
3

[ > SurfaceInt(x^2, [x,y,z]=Sphere(<0,0,0>, 1));

4π
3

[ > SurfaceInt(z, [x,y,z]=Box(0..1, 0..2, 0..3));

33
```

## Calculating a Flux Integral

The **Flux** command in the **VectorCalculus** package can be used to calculate surface integrals of vector fields, or flux integrals. Note the slight difference in defining the surface versus the **SurfaceInt** command - this is because the **VectorField** command already defines the coordinate system.

```
> restart: with(VectorCalculus):
```

```
Warning, the assigned names <,> and <|> now have a global binding
```

```
Warning, these protected names have been redefined and unprotected: *, +, -, ., D, Vector, diff, int, limit, series
```

```
> SetCoordinates('cartesian'[x,y,z]);
```

*cartesian*<sub>x,y,z</sub>

```
> Flux(VectorField(<y,x,z>), Surface(<r*cos(t),r*sin(t),1-r^2>, r=0..1, t=0..2*Pi));
```

$\frac{\pi}{2}$

```
> Flux(VectorField(<z,y,x>), Sphere(<0,0,0>, 1));
```

$\frac{4\pi}{3}$

## Lesson 13-7

### Verifying Stoke's Theorem

Use the **LineInt** and **Flux** commands in the **VectorCalculus** package to verify Stoke's Theorem.

```
> restart: with(VectorCalculus):
```

```
Warning, the assigned names <,> and <|> now have a global binding
```

```
Warning, these protected names have been redefined and unprotected: *, +, -, ., D, Vector, diff, int, limit, series
```

```
> fld:=VectorField(<-y^2,x,z^2>, 'cartesian'[x,y,z]);
```

$fld := -y^2 \bar{e}_x + x \bar{e}_y + z^2 \bar{e}_z$

```
> LineInt(fld, Path(<cos(t), sin(t), 2-sin(t)>, t=0..2*Pi));
```

$\pi$

```
> Flux(Curl(fld), Surface(<r*cos(t),r*sin(t),2-r*sin(t)>, r=0..1, t=0..2*Pi));
```

$\pi$

## Lesson 13-8

## Verifying the Divergence Theorem

Use the `commaFluxnd` in the `VectorCalculus` package along with the `int` command to verify the Divergence Theorem.

```
> restart: with(VectorCalculus):
```

```
Warning, the assigned names <,> and <|> now have a global binding
```

```
Warning, these protected names have been redefined and unprotected: *, +, -, ., D, Vector, diff, int, limit, series
```

```
> fld:=VectorField(<z,y,x>, 'cartesian'[x,y,z]);
```

$$fld := z \bar{e}_x + y \bar{e}_y + x \bar{e}_z$$

```
> Flux(fld, Sphere(<0,0,0>, 1));
```

$$\frac{4\pi}{3}$$

```
> int(int(int(Divergence(fld), z=-sqrt(1-x^2-y^2)..sqrt(1-x^2-y^2)),  
y=-sqrt(1-x^2)..sqrt(1-x^2)), x=-1..1);
```

$$\frac{4\pi}{3}$$

```
>
```

```
>
```