# Means and Moments and Exploring New Plotting Techniques

*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: Extend the concept of the moments of a density function of a solid to applications in probability and engineering.

What do means and multiple integrals have to do with probabilities? How can the method of moments be used to help determine whether or not an object will float in an upright position? These questions will be answered as you explore this module. You will also get to practice new and interesting ways to plot functions.

## 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 sut down *Maple* and start it up again.

## Part I: Masses and Moments Applied to Probability Functions

In multivariable probability, probability density functions are defined over specified regions (domains) in the same way that as density functions are defined over regions of a solid. These probability functions are designed so that the total "mass" is always 1. Probabilities are then

defined to be the integrals of the density functions over a particular subregion of the domain. Consider the following example.

Suppose that a national fast-food outlet is interested in the joint behavior of the random variables *x*, defined as the total time between a customer's arrival at the store and departure from the service window, and *y*, the time that a customer waits in line before reaching the service window. Since x contains the time a customer waits in line, *x* must be greater than *y*. The relative frequency distribution of observed values of *x* and *y* can be modeled by the probability density function:

$f(x, y) = .4e\text{-}1\ e^{(-.1\,x - .3\,y)}$

for $(0 \le y) \le x \quad < \infty$ and 0 elsewhere, where *x* and *y* are measured in minutes.

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

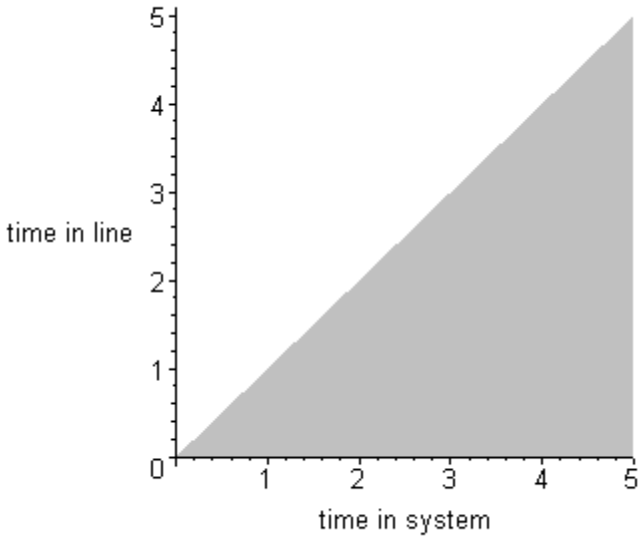
Warning, the name changecoords has been redefined

```
> f:=0.04*exp(-.1*x-.3*y);
```

$$f := 0.04\ e^{(-0.1\,x - 0.3\,y)}$$

We examine the domain of this function. First read in the **plots** package to assist us in graphing.

```
> plot(x, x=0..5, labels=[`time in system`, `time in line`], filled=true, color=grey);
```




We begin by verifying that the integral of the density function over the entire domain is 1.

```
> round(int(int(f, y=0..x), x=0..infinity));
```

$$1$$

To compute the average or expected values of $x$ and $y$ over the domain, we find the first moments of the density function with respect to the $x$ and y axes. The second integral is difficult to evaluate as $x$ approaches $\infty$ , so we substitute a large value of $y$ for the upper bound.

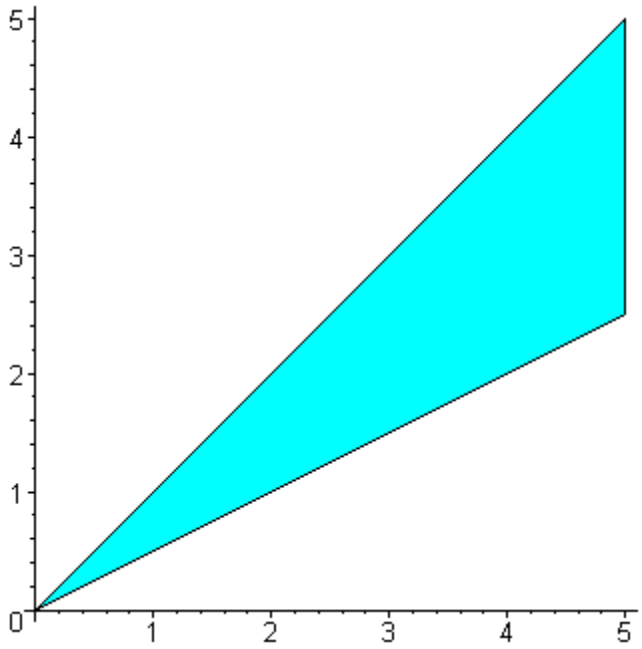
```
> time1:=int(int(x*f, y=0..x), x=0..infinity):
  print( `Expected time in system = `, time1, `minutes`);
  time2:=int(int(y*f, y=0..x), x=0..1000000):
  print( `Expected time waiting in line = `, time2, `minutes`);
```

$$\textit{Expected time in system} = , 12.50000000, \textit{minutes}$$

$$\textit{Expected time waiting in line} = , 2.500000000, \textit{minutes}$$

Now we compute the probability that $y$ is at least one-half as large as $x$. First we can draw the region over which we will integrate (the part for $x < 5$).

```
> polygonplot([[0,0],[5,5],[5,5/2],[0,0]],color=COLOR(RGB,0,1,1));
```



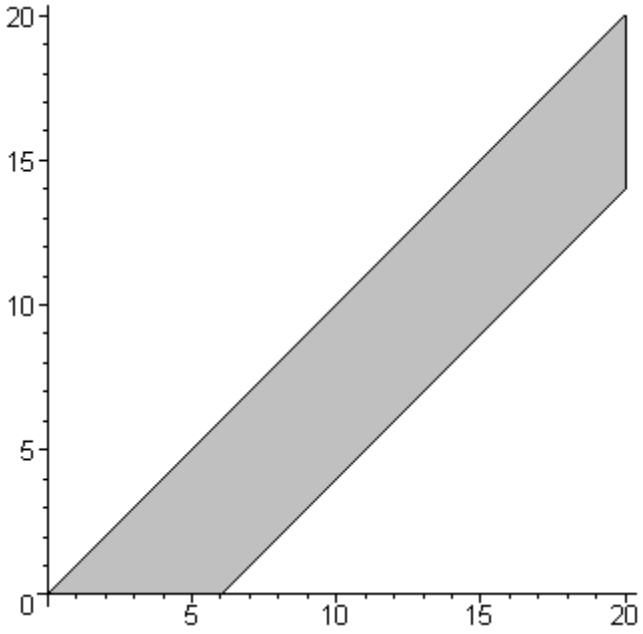

The probability of this event is the integral of the density function over this region.

```
> prob1:=evalf(int(int(f,y=.5*x..x),x=0..infinity)):
  print(`The probability that the time in line is at least half as long as the time in the system i
  `,prob1);
```

*The probability that the time in line is at least half as long as the time in the system is , 0.20000000*

If we want to know the probability that a customer spends no more than six minutes at the service window itself, we represent that quantity by *x* - *y*, and we integrate over the portion of the domain where $x - y \le 6$.

```
> polygonplot([[0,0],[20,20],[20,14],[6,0]],color=grey);
```




Looking at the domain, you can see that it is easier here to integrate with respect to $x$ first.

```
> prob2:=int(int(f,x=y..y+6),y=0..infinity):print(`The probability that the time spent at the se
  desk is no more than 6 minutes is`, prob2);
```

*The probability that the time spent at the service desk is no more than 6 minutes is,* 0.4511883639

# You Try It: Part I

Try your hand at computing the probability that the time at the service desk will be at least 8 minutes. Change the entries to the appropriate terms.

```
> prob3:=int(int(f,x=y..y+6),y=0..infinity):
  print(`The probability that the time spent at the service desk is no more than 6 minutes is`,
  prob3);
```

*The probability that the time spent at the service desk is no more than 6 minutes is,* 0.4511883639

Compute the probability that the time in line is no more than 30% of the time in the system. Change the appropriate terms.

```
> prob4:=int(int(f,y=.5*x..x),x=0..infinity):
  print(`The probability that the time in line is half as long as the time in the system is`, prob
```

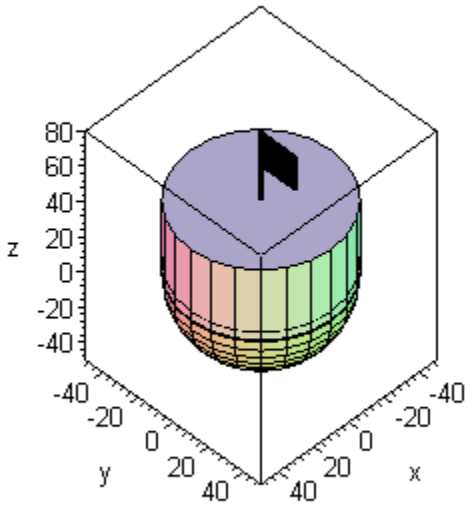
*The probability that the time in line is half as long as the time in the system is,* 0.2000000000

# Part II: Using Moments to Determine Acceptable Heights

**Section 15.2, Extension of Exercise 43**

Suppose that you have a buoy in the form of a solid hemisphere of radius 40 centimeters and uniform density that is topped with a solid cylinder of the same material and radius 40 centimeters. This buoy floats with the center of the hemisphere above the water surface. Determine the maximum height that the cylinder may attain that permits the buoy to continue floating upright. To picture this, load two packages, and plot the shape of the body.

```
> hemisplot:=sphereplot(40, r=0..2*Pi, theta=0..Pi, labels=["x","y","z"], view = [-1..1,-1..1,-1
  axes=boxed):
  pcyl:=plottools[cylinder]([0,0,1],40,40):
  flagpole:=plot3d([0,0,z], z=35..80,x=-10..10,axes=boxed, thickness=3):
  flag:=plot3d([0,u,v], u=0..20, v=60..80, axes=boxed):
  display({hemisplot, pcyl, flagpole, flag}, view=[-50..50, -50..50, -50..80],
  scaling=CONSTRAINED);
```




You may assume that the flag on top is weightless.

The method of moments can help us determine that the buoyancy force exerted by the body of water due to the amount of water displaced will be in a direction through the point that would be the midpoint of the sphere, as long as the object is upright or tilted so that no part of the cylinder is immersed. Hence, if the center of gravity of the solid is at that point or closer to to the bottom of the hemisphere, the object will stay upright. If the center of gravity is in the region of the cylinder, the object will tilt.

We first determine the z-coordinate of the center of gravity of the hemisphere. Recall that the volume of a hemisphere with radius of $r$ is $\frac{2\pi r^3}{3}$ .

```
> volhemis:=2/3*Pi*40^3:
  hemis:=-sqrt(1600-r^2):
  unassign('r,z,theta');
  zbar:=int(int(int(r*z,z=hemis..0), r=0..40),theta=0..2*Pi)/volhemis;
```

$$zbar := -15$$

The z-coordinate of the center of mass of the cylinder is h/2, so if we equate the first moment of the hemisphere to the first moment of the cylinder, we can solve for h. Since the radius is 40 here, the volume of the cylinder is 1600*Pi* h.

```
> height:=solve(-zbar*volhemis=h/2*(1600*Pi*h),h);
```

$$height := 20\sqrt{2}, -20\sqrt{2}$$

Therefore, the height can be at most 20 times the $\sqrt{2}$ . In general, if the radius were R, the maximum height of the cylinder could be at most R times the $\sqrt{2}$ / 2.

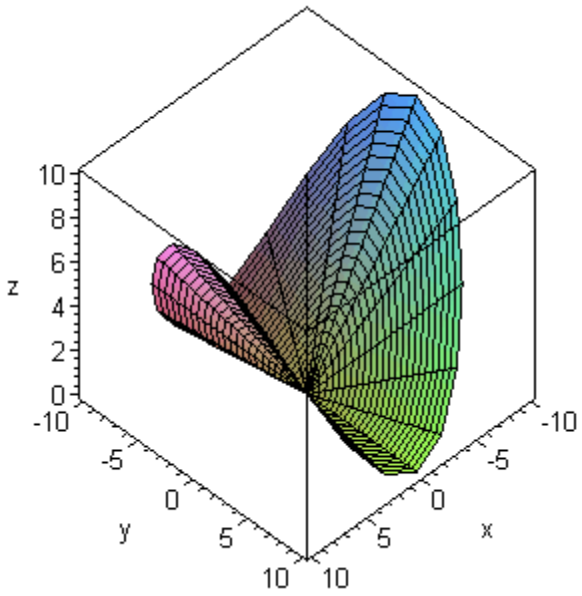
# You Try It: Part II

Experiment with some of the 3-D plot functions presented in this section. First just practice with a cylindrical plot. In the following example, $z$ is a function of $r$ and $\theta$ . The first argument of **cylinderplot( )** is a list of the three cylindrical coordinates of points on the surface, and we express the $z$-coordinate as a function of the first two, $r$ and $\theta$ . Try some of your own functions.

Change the required terms.

```
> unassign('theta','r');
  z:=r*(cos(theta))^2;
  cylinderplot([r,theta,z], r=0..10, theta=0..2*Pi,labels=["x","y","z"],axes=boxed);
```

$$z := r\cos(\theta)^2$$



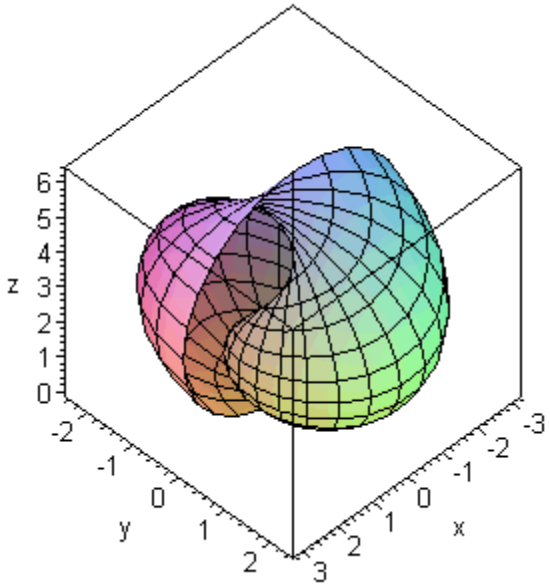

Try it out with some other functions.

You can create spherical plots with the same package you used to create cylindrical plots. In these plots, it is assumed that $\rho$ is a function of $\phi$ and $\theta$. Try some of your own by changing the terms in the function $\rho$. Not everything will give satisfying results.

> **rho:=theta*cos(phi);**

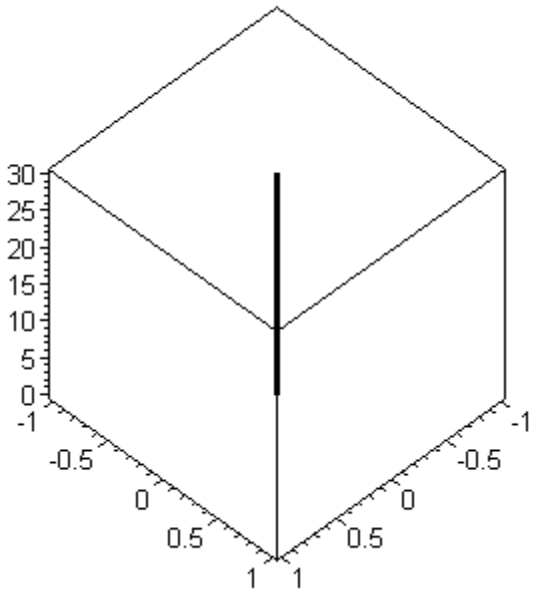
$$\rho := \theta \cos(\phi)$$

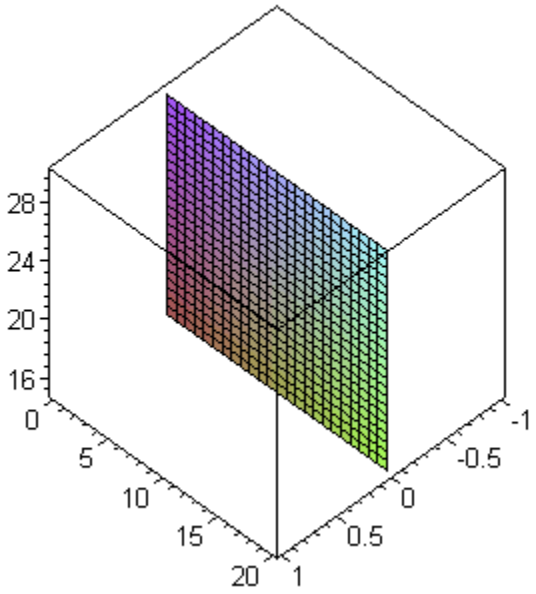
> **sphereplot(rho, theta=0..2*Pi, phi=0..Pi, labels=["x","y","z"],axes=boxed);**

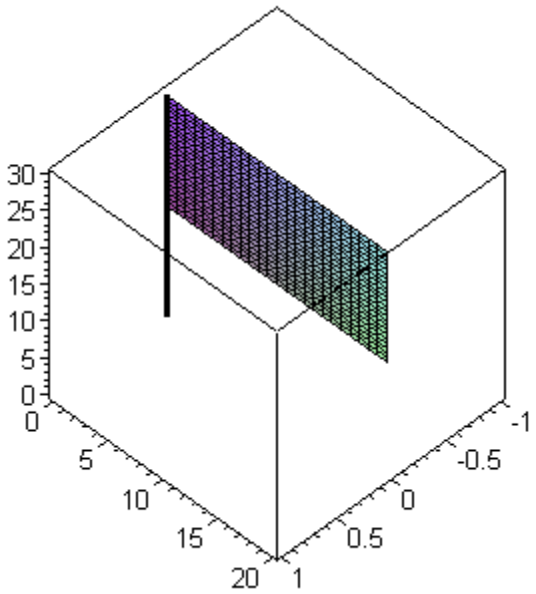



You may have noticed that to draw the flag we had to first generate points and then plot the points. The difference between the two was that the pole could be thought of as a line, whereas, the flag was more like a surface. Experiment with the following commands and draw something else by changing the terms in the plot commands. Remember to leave one parameter in the first expression and two parameters in the second experession.

```
> unassign('z');
  newflagpole:=plot3d([0,0,z], z=0..30,x=-10..10,axes=boxed, thickness=3):
  print(newflagpole);
  newflag:=plot3d([0,u,v], u=0..20, v=15..30, axes=boxed):
  print(newflag);
  print(display({newflag, newflagpole}));
```


30
25
20
15
10
5
0
-1
-0.5
0
0.5
1
1
0.5
0
-0.5
-1


28
24
20
16
0
5
10
15
20
1
0.5
0
-0.5
-1


30
25
20
15
10
5
0
0
5
10
15
20
1
0.5
0
-0.5
-1

>