APLICACIONES DEL CAPÍTULO V.

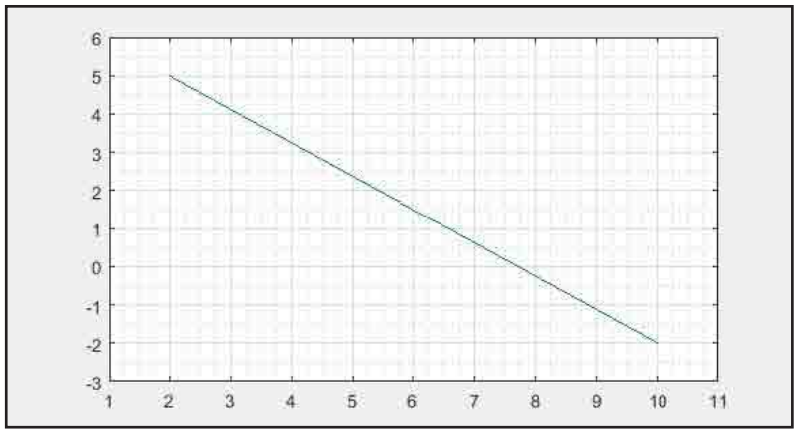
5.1. Elaborar un programa para graficar una recta cuyas coordenadas (x1;y1) y (x2;y2) se deben ingresar desde el teclado (no olvide que para hacer un programa tiene que abrir el editor de programas presione ctrl+N).

El programa sería.

```
%Programa para graficar una recta de (x1;y1) hasta (x2;y2)
clc;
x1=input('Ingrese x1: ');
y1=input('Ingrese y1: ');
x2=input('Ingrese x2: ');
y2=input('Ingrese y2: ');
x=[x1,x2];
y=[y1,y2];
plot(x,y);
axis([min(x)-1,max(x)+1,min(y)-1,max(y)+1]);
grid on;
grid minor;
```

Cuando se ejecuta el programa para graficar una recta desde (2;5) a (10;-2), el resultado es el siguiente.

```
Ingrese x1: 2
Ingrese y1: 5
Ingrese x2: 10
Ingrese y2: -2
```



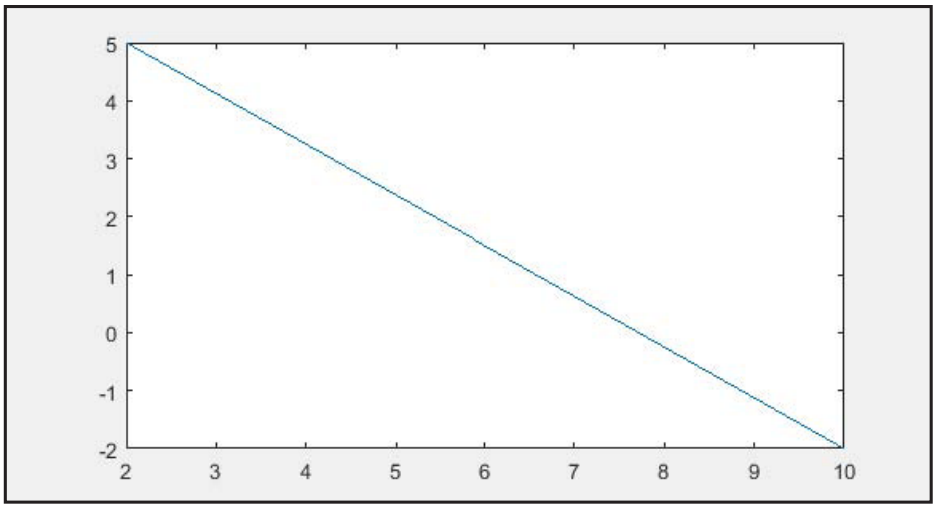
5.2 Diseñar una función para graficar una recta cuyas coordenadas (x1;y1) y (x2;y2) se deben ingresar como parámetros de la función.

La función sería.

```
function []=recta(x1,y1,x2,y2)
      %Función para graficar una recta de (x1;y1) hasta (x2;y2)
      x=[x1,x2];
      y=[y1,y2];
      plot(x,y);
end
```

Cuando se ejecuta la función para graficar una recta desde (2;5) a (10;-2), sería:

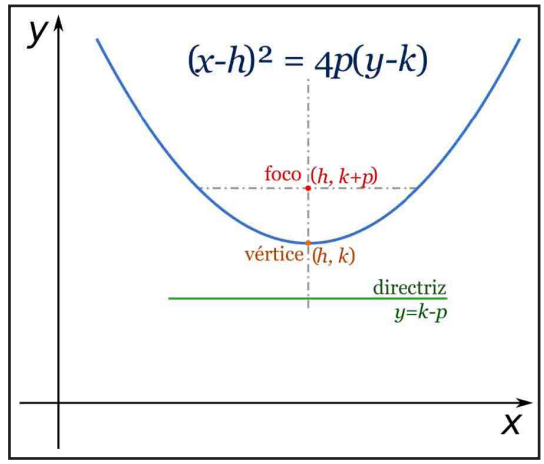
```
>> recta(2,5,10,-2)
```



Nota. A diferencia de un programa, en una función normalmente no se debe utilizar sentencias ***axis, grid, text***, etc.; si esas sentencias son necesarias se debe hacer desde le ***prompt*** (>>) tal como se muestra a continuación.

```
>> recta(2,5,10,-2) %ejecución de la función.
>> grid on;
>> grid minor;
>> axis([1,11,-3,6]);
>> text(2.5,5,'(2;5)');
>> text(10,-2,'(10;-2)');
>> title('Gráfico de una recta');
```

5.3. Elaborar un programa para graficar una parábola con vértice en el punto (h;k) y el foco esté ubicado a una distancia "p" del vértice (ver siguiente figura); los valores de h, k y p se deben ingresar desde el teclado.



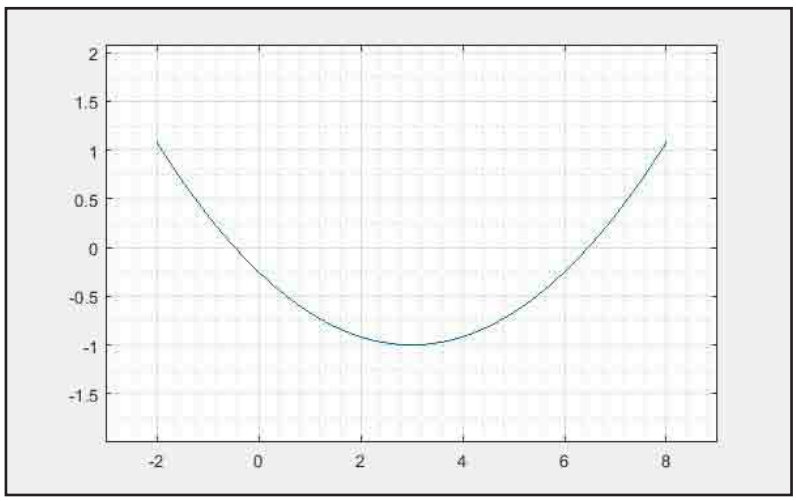
El programa sería.

```
%Programa para graficar una parábola con vértice (h;k) y foco.
clc;
clear x y;
h=input('Ingrese la abscisa del vértice (h): ');
k=input('Ingrese la ordenada del vértice (k): ');
p=input('Ingrese distancia del vértice al foco (p): ');
x=linspace(h-5,h+5,100);
y=(x-h).^2/(4*p)+k; %se despeja y de la forma general
plot(x,y);
grid on;
```

```
grid minor;
axis([min(x)-1,max(x)+1,min(y)-1,max(y)+1]);
```

Cuando se ejecuta el programa para graficar una parábola con vértice en el punto (3;-1) y el foco ubicado a una distancia 3 del vértice, el resultado sería:

```
Ingrese la abscisa del vértice (h): 3
Ingrese la ordenada del vértice (k): -1
Ingrese distancia del vértice al foco (p): 3
```



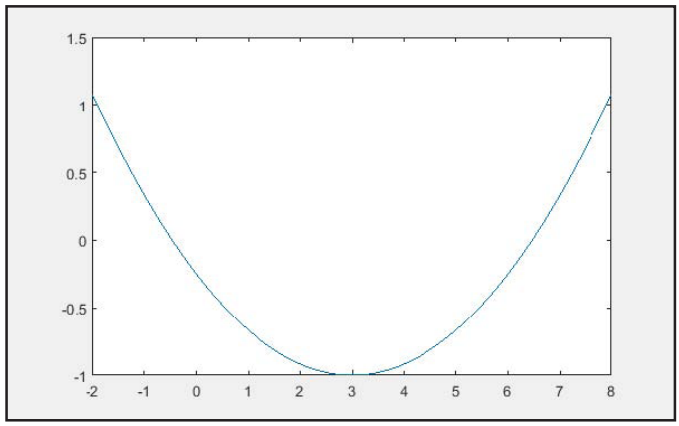
5.4. Diseñar una función para graficar una parábola del ejemplo anterior con vértice en el punto (h;k) y el foco esté ubicado a una distancia "p" del vértice; los datos h, k y p deben ser .

La función sería.

```
function []=parabola(h,k,p)
      %Programa para graficar una parábola con vértice (h;k)
      % y distancia al foco (p)
      x=linspace(h-5,h+5,100);
      y=(x-h).^2/(4*p)+k;
      plot(x,y);
end
```

Cuando se ejecuta la función parábola con vértice(3;-1) y foco igual a 3, sería:

```
>> parabola(3,-1,3)
```



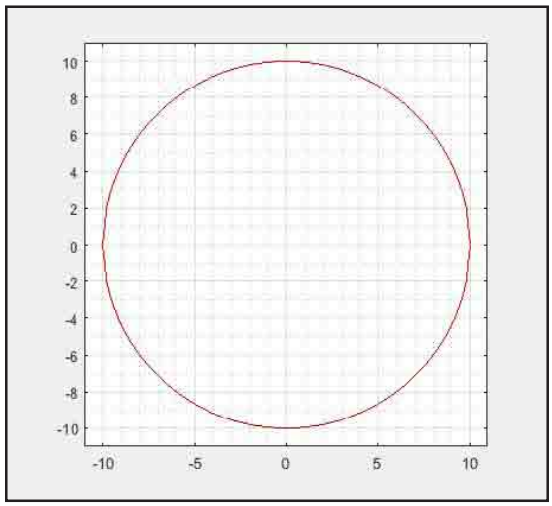
5.5. Elaborar un programa para graficar una circunferencia de color rojo con centro en el origen de coordenadas, cuyo radio se debe ingresar desde el teclado.

El programa sería.

```
%Programa para graficar un circunferencia de radio y color rojo.
clc;
radio=input('Ingrese radio: ');
x=linspace(-radio,radio,100);
y=sqrt(radio.^2-x.^2);
plot(x,y,x,-y,'color','r')
axis equal;
axis([-radio-1,radio+1,-radio-1,radio+1]);
grid on;
grid minor;
```

Cuando se ejecuta el programa con radio=10, el resultado es el siguiente.

```
Ingrese radio: 10
```



Otra forma de programar una circunferencia sería el siguiente:

```
%Programa para graficar un circunferencia de radio y color rojo.
clc;
radio=input('Ingrese radio: ');
t=linspace(0,2*pi,100);
x=radio*cos(t);
y=radio*sin(t);
plot(x,y,'-r')
axis equal axis([-radio-1,radio+1,-radio-1,radio+1]);
grid on
grid minor
```

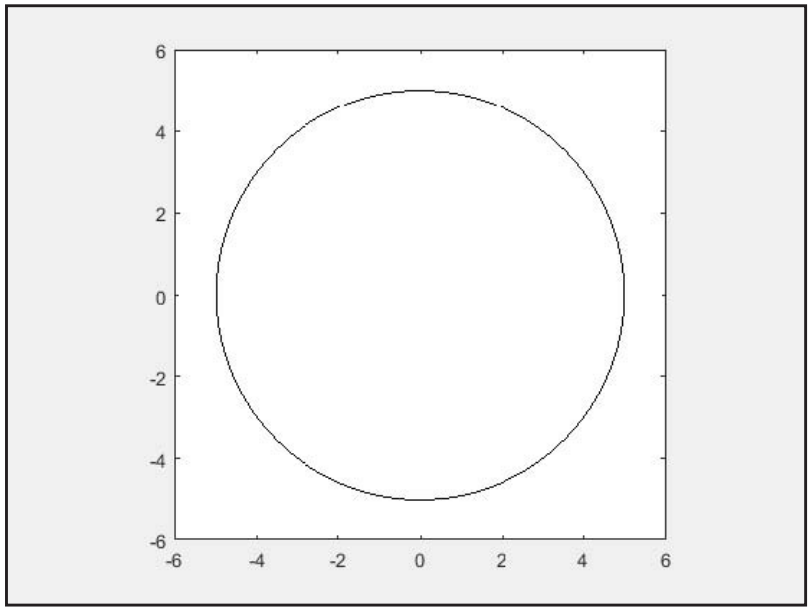
5.6. Diseñar una función para graficar una circunferencia del ejemplo anterior con centro en el punto (0;0) cuyo radio y color se debe ingresar como parámetros de la función. (Nota: el color se debe ingresar con los mismos caracteres de la función ***plot*** que se muestra en el Cuadro 5.1).

La función sería.

```
function []=circunferencia(radio,color)
      %Función para graficar una circunferencia de radio y color.
      %El color se ingresa tipo cadena, por ejemplo 'r' de rojo
      t=linspace(0,2*pi,100);
      x=radio*cos(t);
      y=radio*sin(t);
      plot(x,y,color);
      axis equal;
end
```

Cuando se ejecuta la función con radio=10 y color negro, sería:

```
>> circunferencia(10,'k')
```



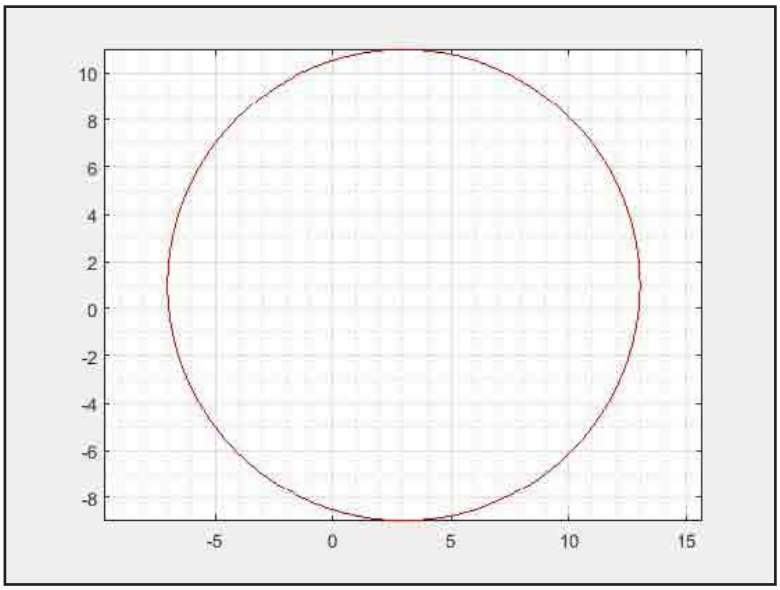
5.7. Diseñar una función para graficar una circunferencia del ejemplo anterior pero con centro en el punto (xc;yc) cuyos valores de xc, yc, radio y color se debe ingresar como parámetros de la función. (Nota: el color se debe ingresar con los mismos caracteres de la función plot que se muestra en el Cuadro 5.1).

La función sería.

```
function []=circunferenciaxy(xc,yc,radio,color)
      %Función para graficar una circunferencia de radio y color.
      %con centro en el punto (xc;yc)
      t=linspace(0,2*pi,100);
      x=xc+radio*cos(t);
      y=yc+radio*sin(t);
      plot(x,y,color);
      axis equal
end
```

Cuando se ejecuta la función con centro en (3;1), radio=10 y color rojo, sería:

```
>> circunferenciaxy(3,1,10,'r')
>> grid on
>> grid minor
```



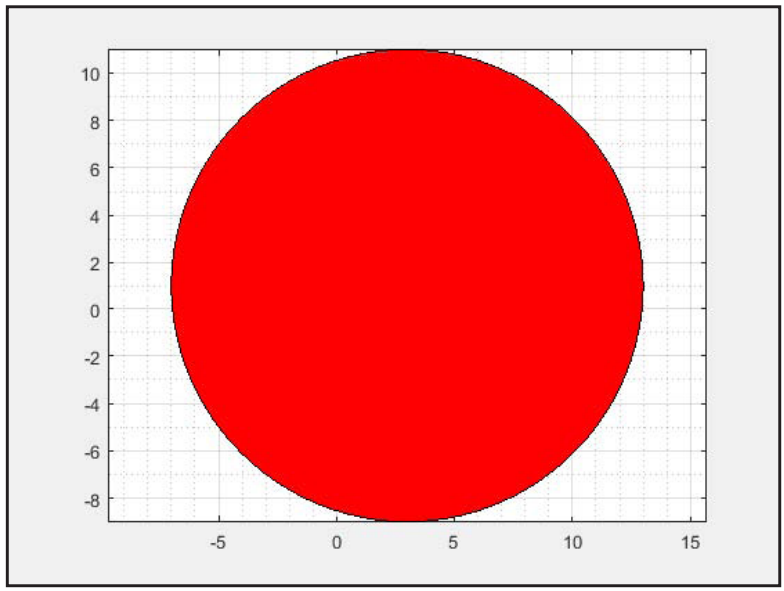
5.8. Diseñar una función para graficar un círculo con centro en el punto (xc;yc) cuyos valores de xc, yc, radio y color se debe ingresar como parámetros de la función. (**Nota**: un círculo es una circunferencia rellenada y se usa la función ***fill***).

La función sería.

```
function []=circuloxy(xc,yc,radio,color)
      %Función para graficar un círculo de radio y color.
      %con centro en el punto (xc;yc)
      t=linspace(0,2*pi,100);
      x=xc+radio*cos(t);
      y=yc+radio*sin(t);
      fill(x,y,color);
      axis equal;
end
```

Cuando se ejecuta la función con centro en (3;1), radio=10 y color rojo, sería:

```
>> circuloxy(3,1,10,'r')
>> grid on
>> grid minor
```



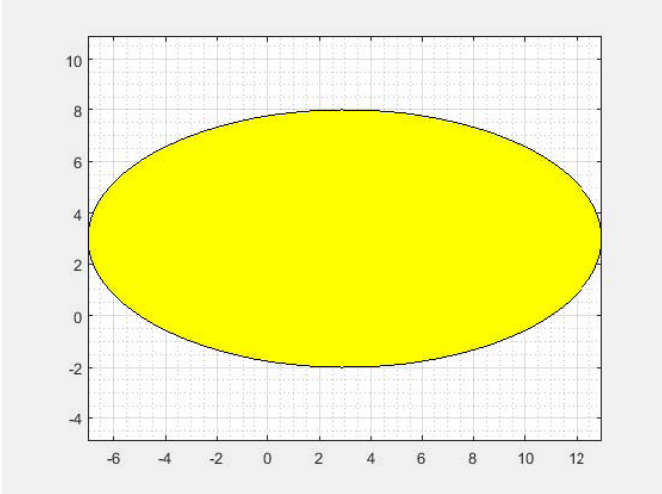
5.9. Diseñar una función para graficar una elipse con relleno con centro en el punto (xc;yc) cuyos valores de xc, yc, radio horizontal y radio vertical y color se debe ingresar como parámetros de la función.

La función sería.

```
function []=elipsexy(xc,yc,radiox,radioy,color)
      %Función para graficar una elipse de radiox y radioy y color.
      %con centro en el punto (xc;yc)
      t=linspace(0,2*pi,100);
      x=xc+radiox*cos(t);
      y=yc+radioy*sin(t);
      fill(x,y,color);
      axis equal;
end
```

Cuando se ejecuta la función con centro en (3;3), radiox=10, radioy=5 y color amarillo, sería:

```
>> elipsexy(3,3,10,5,'y');
>> grid on;
>> grid minor;
```

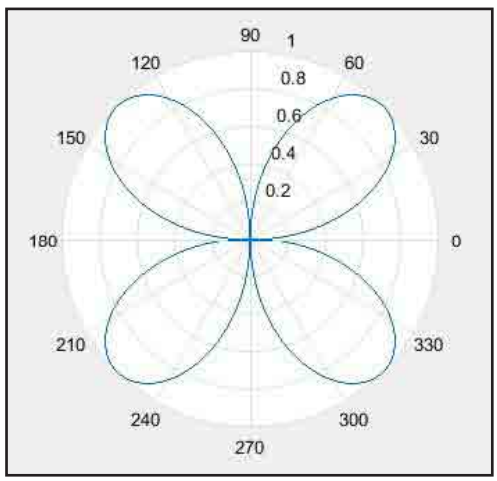


5.10. Diseñar una función para graficar “n” hojas en un plano polar.

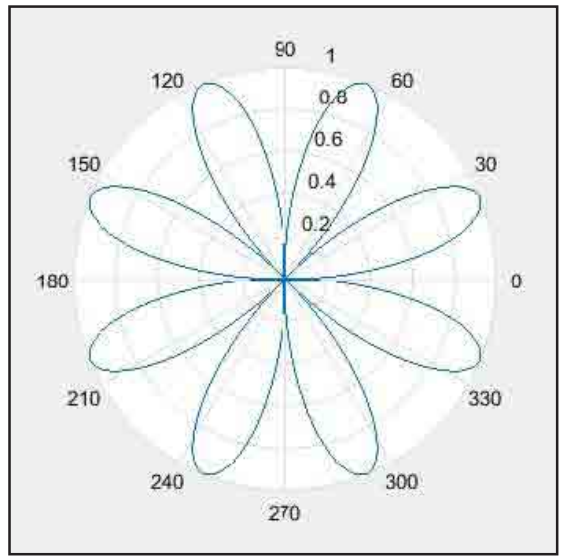
La función sería.

```
function []=hojas(n)
      %Función para grafica 'n' hojas en un plano polar
      t=linspace(0,2*pi,1000);
      r=sqrt(cos(n*t));
      polar(t,r);
end
```

Cuando se ejecuta la función con 4 hojas sería:



Cuando se ejecuta la función con 8 hojas sería:

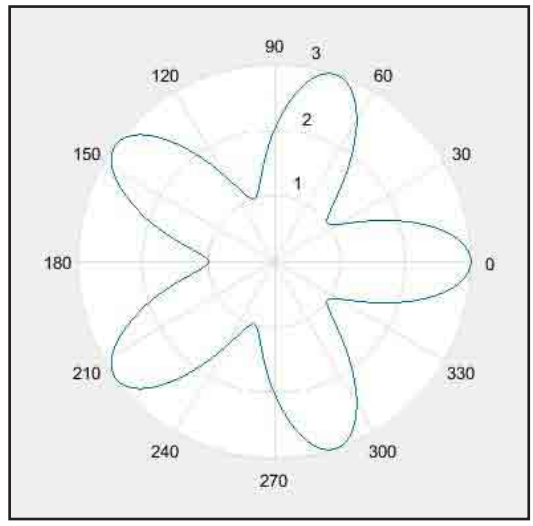


5.11. Diseñar una función para graficar una rosa con "n" pétalos en un plano polar.

La función sería.

```
function []=rosa(n)
      %Función para grafica 'n' pétalos de rosa en un plano polar
      t=linspace(0,2*pi,1000);
      r=2+cos(n*t);
      polar(t,r);
end
```

Cuando se ejecuta la función con 5 pétalos sería:

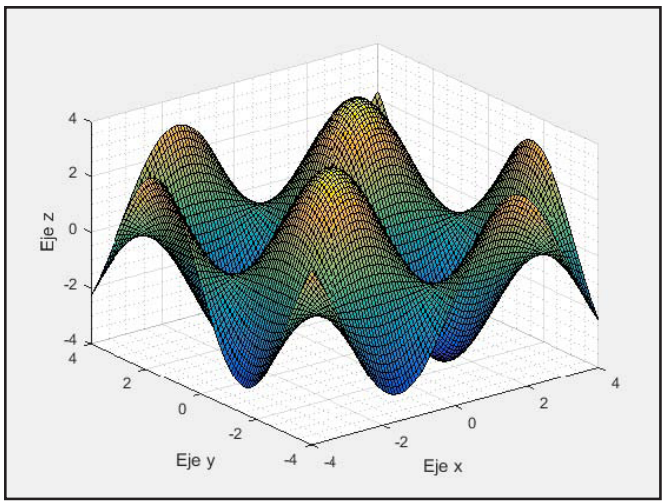


5.12. Diseñar una función para simular la gráfica de cerros con "n" cerros.

La función sería.

```
function []=cerros(n)
      Función para graficar 'n' cerros en 3D
      t=linspace(-n,n,100);
      [x,y]=meshgrid(t);
      z=n.*sin(x).*sin(y); surf(x,y,z);
      xlabel('Eje x');
      ylabel('Eje y');
      zlabel('Eje z');
      grid minor
end
```

Cuando se ejecuta la función con 4 cerros sería:



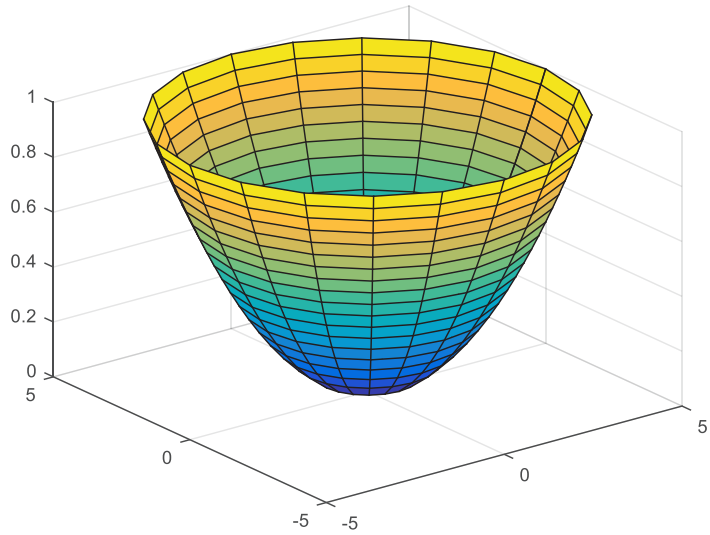
5.13. Diseñar una función para graficar un paraboloide con radio en la base (rmenor) y radio en la salida (rmayor) dichos valores se deben ingresar como parámetros de la función.

La función sería.

```
function []=paraboloide(rmenor,rmayor)
      %Función para graficar un paraboloide rmenor y rmayor.
      t=linspace(rmenor^2,rmayor^2,20);
      r=sqrt(t);
      cylinder(r);
end
```

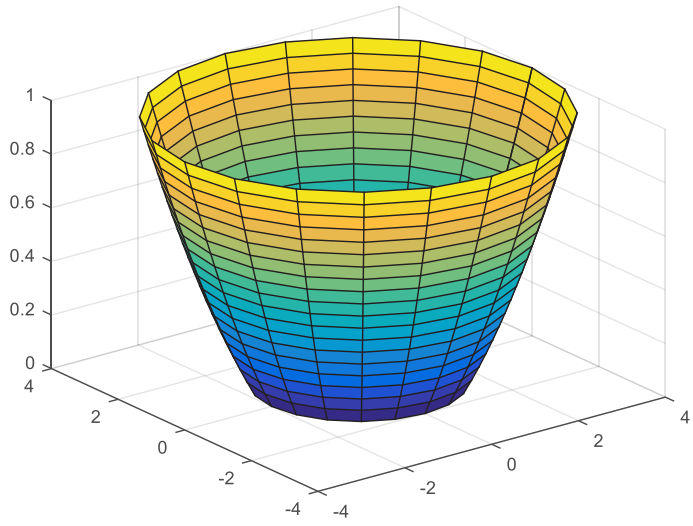
Cuando se ejecuta la función con radio menor=0 y radio mayor=5 sería:

```
>> paraboloide(0,5);
```



Cuando se ejecuta la función con radio menor igual=2 y radio mayor=4 sería:

```
>> paraboloide(2,4);
```



5.14. Diseñar una función que devuelva el tiempo y la altura del cual se debe lanzar un objeto, además la función debe graficar el movimiento de una partícula que se lanza con una velocidad inicial (vo), el ángulo (ang) y a una distancia (dx) cuyos datos deben ser los parámetros de la función.

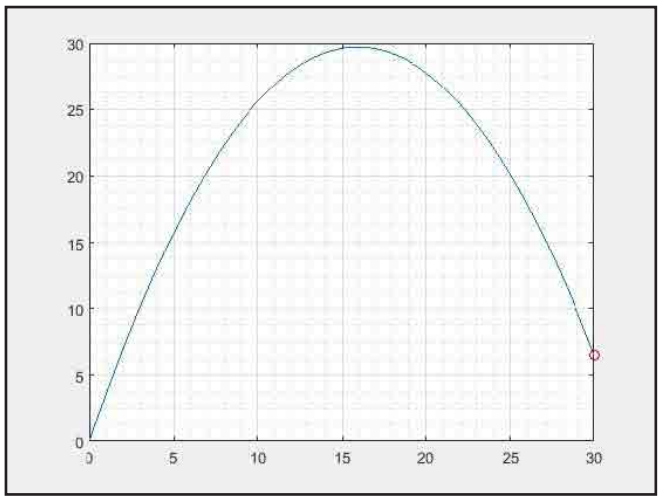
La función sería.

```
function [tiempo,h]=lanzamientoproyectil(vo,ang,xd)
      %Función que grafica el lanzamiento de un proyectil
      %Se debe ingresar la veloc. Inicial (vo) el ángulo (ang) en
      %grados sexagesimales y la distancia horizontal (dx)
      g=9.81; %gravedad
      tiempo=xd/(vo*cosd(ang));
      t=linspace(0,tiempo,100);
      x=vo*cosd(ang)*t;
      y=vo*sind(ang)*t-g*t.^2/2;
      h=y(end);
      for (i=1:100)
            plot(x,y,x(i),y(i),'or');
            pause(0.025);
      end
end
```

Cuando se ejecuta la función con vo=25 m/s, ang=75° y xd=30 m sería:

```
>> [t,h]=lanzamientoproyectil(25,75,30)
t =
  4.6364
h =
  6.5206
```

Y el gráfico se muestra en la siguiente figura.



5.15. Diseñar una función que genere las matrices x, y, z que permitan graficar un ***torus***, sabiendo que para generarla se usa las siguientes fórmulas:

x = r.cos(θ)

y = r.sin(θ)

$$z = \pm \sqrt{a^2 - (\sqrt{x^2 + y^2} - b)^2}$$

Donde: b-a ≤ r ≤ b+a , 0 ≤ θ ≤ 2π y b > a.

La función sería.

```
function [X,Y,Z]=matricestoroide(a,b)
      %Función que genera las matrices para graficar un toroide
      r=linspace(b-a,b+a,10);
      teta=linspace(0,2*pi,20);
      x=r'*cos(teta); %multiplica la transpuesta de r por cos(teta)
      y=r'*sin(teta); %multiplica la transpuesta de r por sin(teta)
      z=sqrt(a^2-(sqrt(x.^2+y.^2)-b).^2);
      X=[x,x];
      Y=[y,y];
      Z=[z,-z];
end
```

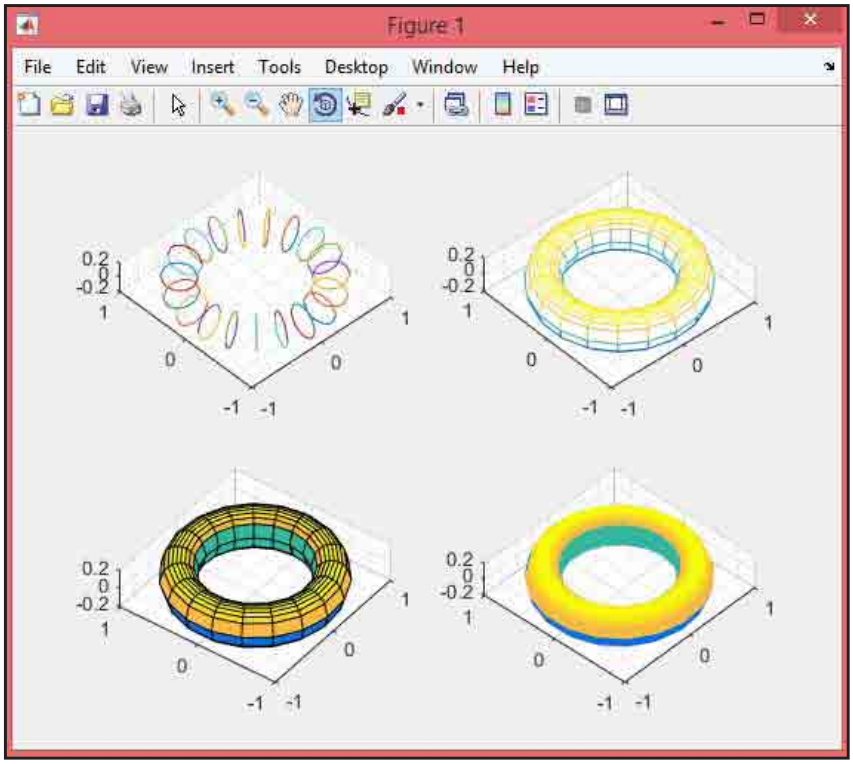
Cuando se ejecuta la función con a = 0,2 y b = 0,8 se obtiene las matrices x, y, z.

```
>> [x,y,z]=matricestoroide(0.2,0.8); %genera las matrices x,y,z.
```

Ahora como ya tenemos los valores de ***x, y*** y ***z***; utilizaremos las funciones ***plot3***, ***mesh*** y ***surf*** en una misma gráfica utilizando la función ***subplot***, tal como se muestra a continuación.

```
>> subplot(2,2,1);plot3(x,y,z);grid minor
>> subplot(2,2,2);mesh(x,y,z);grid minor
>> subplot(2,2,3);surf(x,y,z);grid minor
>> subplot(2,2,4);surf(x,y,z),shading flat;grid minor
```

El resultado se muestra en la siguiente figura.



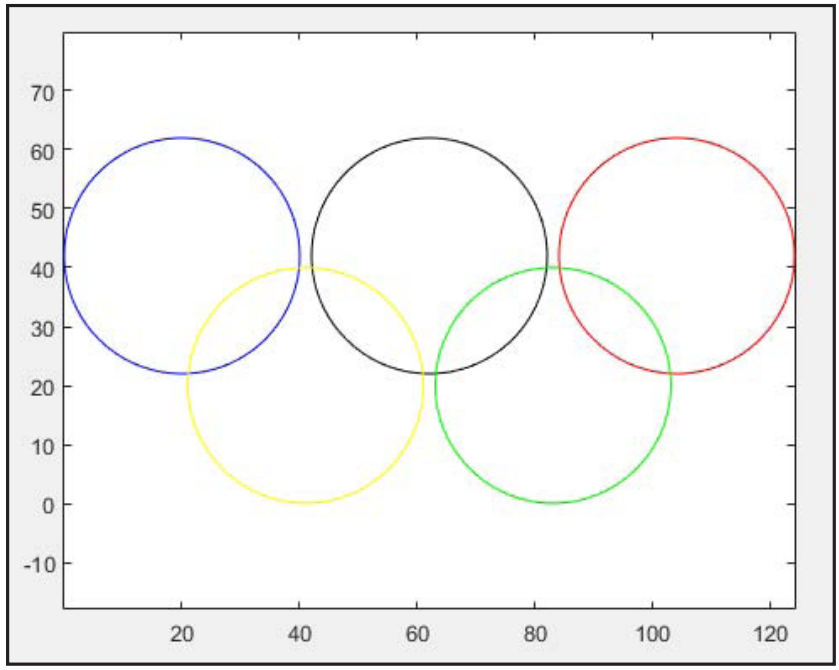
5.16. Usando la función **circunferenciaxy** diseñado en el ejercicio 5.7, elaborar un programa para graficar los aros olímpicos con sus respectivos colores.

```
%Programa para graficar los aros olímpicos con sus colores
clc
r=input('Ingrese radio: ');
circunferenciaxy(r,2*r+r/10,r,'b') %aro azul
axis equal
hold on
circunferenciaxy(3*r+r/10,2*r+r/10,r,'k') %aro negro
circunferenciaxy(5*r+r/5,2*r+r/10,r,'r') %aro rojo
circunferenciaxy(2*r+r/20,r,r,'y') %aro amarillo
circunferenciaxy(4*r+3*r/20,r,r,'g') %aro verde
```

Cuando se ejecuta el programa con r = 20 sería:

```
Ingrese radio: 20
```

Y el gráfico se muestra en la siguiente figura.



5.17. Diseñar una función que grafique un corazón, para ello se debe graficar la siguiente función:

$$f(x) = (\sqrt{\cos(x)}.\cos(250x) + \sqrt{|x|} - 0{,}7) \cdot (\sqrt[100]{4 - x^2})$$

Para 5000 puntos de x en el intervalo de [-2,2].

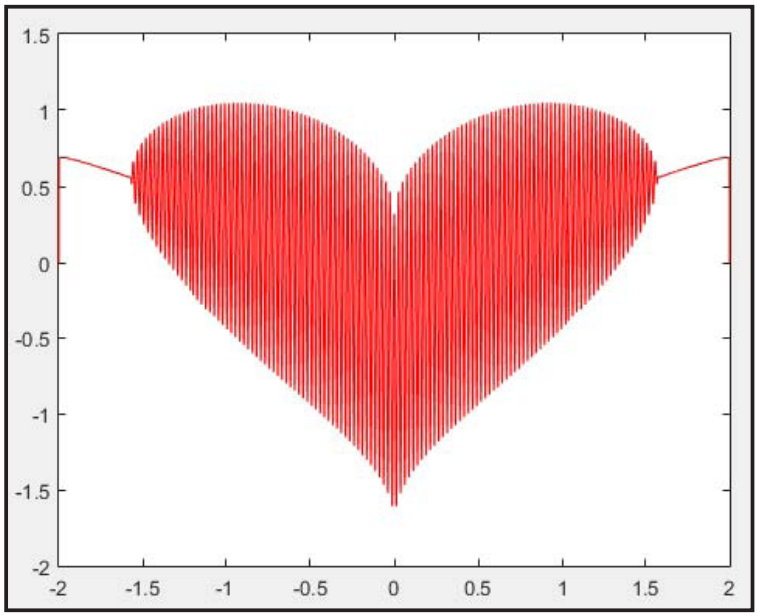
La función sería.

```
function []=corazon()
      %Función que grafica un corazón.
      x=linspace(-2,2,5000);
      y=(sqrt(cos(x)).*cos(250*x)+sqrt(abs(x))-0.7).*(4-x.^2).^0.01;
      plot(x,y,'r');
end
```

Cuando se ejecuta la función.

```
>> corazon
```

Se obtiene la siguiente figura.



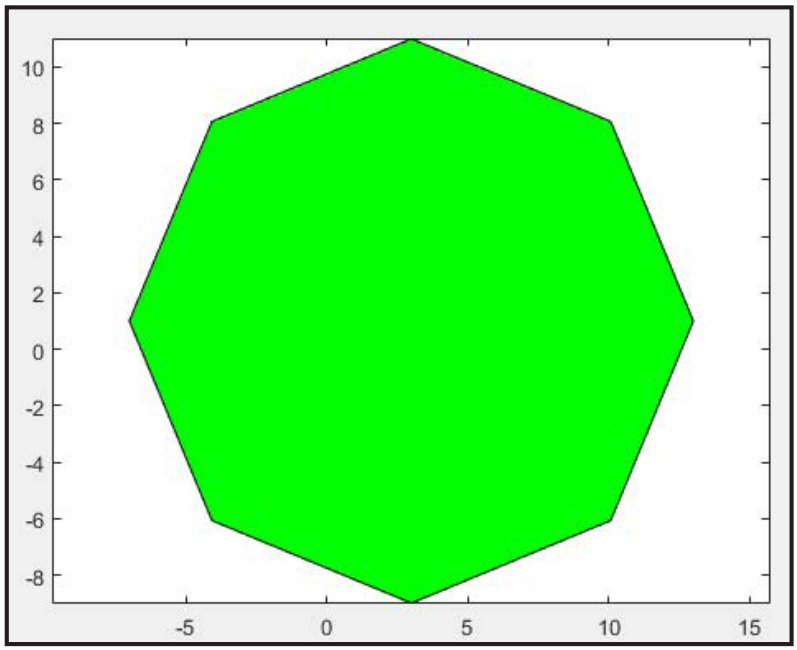
5.18. Diseñar una función para graficar un polígono de "n" lados inscrito dentro de un círculo con centro en el punto (xc;yc) cuyos valores de xc, yc, radio, número de lados y color se debe ingresar como parámetros de la función. (Nota: el polígono debe ser rellenada con el color indicado, por tanto, de debe usar la función ***fill***).

La función sería.

```
function []=poligono(xc,yc,radio,n,color)
      t=linspace(0,2*pi,n);
      x=xc+radio*cos(t);
      y=yc+radio*sin(t);
      fill(x,y,color); axis
      equal;
end
```

Cuando se ejecuta la función con centro en (3;1), radio=10, 8 lados y color verde, sería:

```
>> poligono(3,1,10,8,'g')
```


10
8
6
4
2
0
-2
-4
-6
-8
-5
0
5
10
15