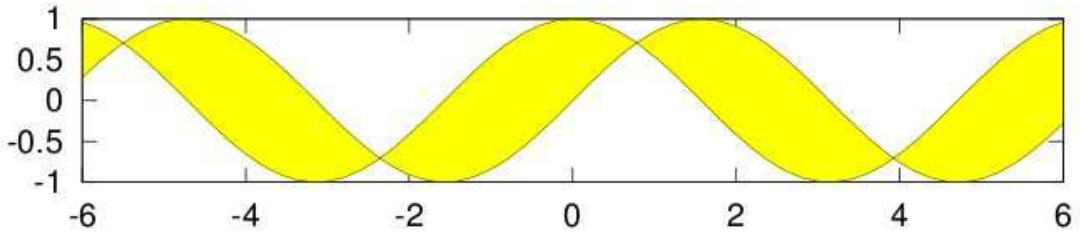

Figura 2: Grafico generado por *region2*

1.6. Creando funciones

En la sección 1.4 se estudió como se puede interactuar con un script usando el comando **input**. Cuando se necesita realizar una operación varias veces para distintos valores de un parámetro resulta más eficiente definir una **función**. Las funciones de *Octave* pueden tener como resultado un número, un vector, un gráfico, etc.

La tercera situación problemática es:

Elaborar una función de octave que sombree para dos funciones $f(x)$ y $g(x)$ dadas cualquiera de las regiones:

- $f(x) > g(x)$
- $f(x) \geq g(x)$
- $f(x) < g(x)$
- $f(x) \leq g(x)$

para cualquier intervalo deseado.

Una solución para el problema se encuentra en la siguiente función:

region4

```
function region4 (xmin,xmax,nx,operador)
x=[xmin:(xmax-xmin)/nx:xmax];
y1=sin(x);
y2=cos(x);
figure
hold on;
axis([min(x),max(x),min([y1,y2]),max([y1,y2])]);
plot(x,y1,'r');
plot(x,y2,'g');
for i=1:columns(x)
        switch operador
        case '>'
        condicion=y1(i)>y2(i);
        case '<'
```

```
            condicion=y1(i)<y2(i);
            case '>='
            condicion=y1(i)>=y2(i);
            case '<='
            condicion=y1(i)<=y2(i);
            endswitch
            if (condicion)
              usleep(1);
              plot([x(i),x(i)],[y1(i),y2(i)],'b');
            endif
end
hold off;

endfunction;
```

En esta función los datos de entrada son el menor y el mayor valor de x la cantidad de subdivisiones del eje x y el operador de comparación.

La sintaxis de del comando function es:

```
function [lista de resultados] = nombre (lista de argumentos)
      acciones
endfunction
```

donde **resultado** es el nombre de una varible o una lista de varibles que al final de la ejecución tienen los resultados esperados.

Observación 2 *La lista de de argumentos y la lista de resultados pueden ser vacías.*

La sintaxis del switch es:

```
switch expresion
      case etiqueta
            acciones
      case etiqueta
            acciones
      ...
      otherwise
            acciones
endswitch
```

Ejemplo 1 *Sombrear la región encerrada entre por los gráficos de* $f(x) = sin(x)$ *y* $g(x) = cos(x)$ *donde* $sin(x) > cos(x)$ *y* $x \in [-6,6]$ *usando 60 subintervalos.*

Ejemplo 2 *Sombrear la región encerrada entre por los gráficos de* $f(x) = sin(x)$ *y* $g(x) = cos(x)$ *donde* $sin(x) < cos(x)$ *y* $x \in [-6,6]$ *usando 60 subintervalos.*

Observe que el cuarto argumento está entre comillas, esto significa que es de tipo **string** (o cadena de caracteres).

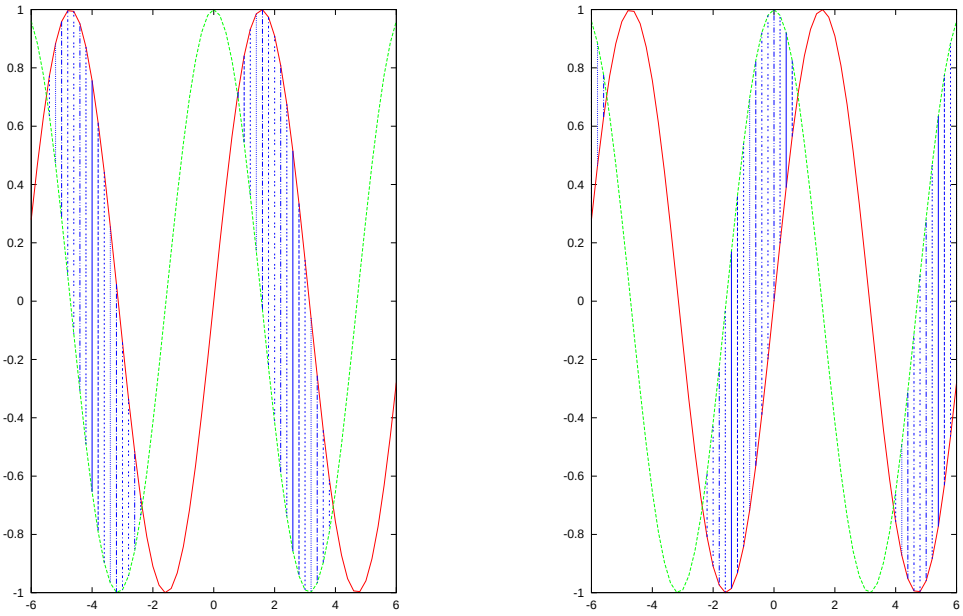

Figura 3: A la izquierda la salida de region4(-6,6,60,'>') y a la derecha la correspondiente a region4(-6,6,60,'<').

Ejemplo 3 *Sombrear la región encerrada entre por los gráficos de* $f(x) = sin(x)$ *y* $g(x) = cos(x)$ *donde* $sin(x) > cos(x)$ *y* $x \in [-60, 60]$ *usando 60 subintervalos.*

En *region4* las funciones a comparar están prefijadas pero con una pequeña modificación es posible pasar las funciones como argumento.

region5

```
function region5 (xmin,xmax,nx,f1,operador,f2)
x=[xmin:(xmax-xmin)/nx:xmax];
y1=feval(f1,x);
y2=feval(f2,x);
figure
hold on;
axis([min(x),max(x),min([y1,y2]),max([y1,y2])]);
plot(x,y1,'r');
plot(x,y2,'g');
for i=1:columns(x)
        switch operador
        case '>'
        condicion=y1(i)>y2(i);
        case '<'
        condicion=y1(i)<y2(i);
        case '>='
```

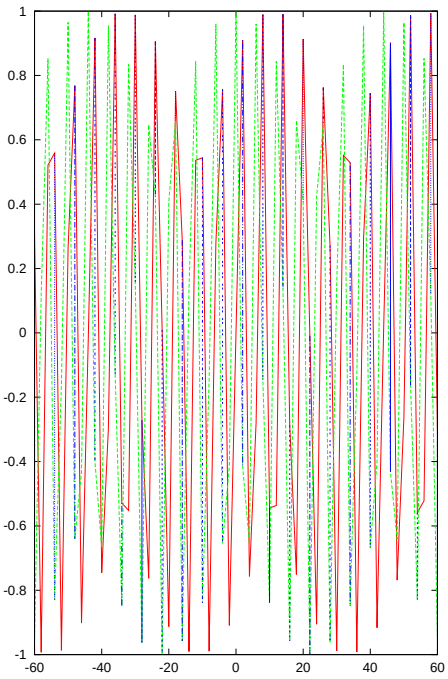

Figura 4: Gráfico correspondiente al ejemplo 3 en el que se observa que *Octave* grafica poligonales.

```
            condicion=y1(i)>=y2(i);
            case '<='
            condicion=y1(i)<=y2(i);
            endswitch
            if (condicion)
              usleep(1);
            plot([x(i),x(i)],[y1(i),y2(i)],'b');
            endif
end
hold off;
endfunction;
```

La líneas:

```
y1=feval(f1,x);
y2=feval(f2,x);
```

le indican a *Octave* que uno de los agumentos debe ser interpretado como función, aúnque en el llamado de la función se lo ingrese como un string.

Ejemplo 4 *Usando la función region5.m sombrear la región encerrada entre por los gráficos de* $f(x) = sin(x)$ *y* $g(x) = cos(x)$ *donde* $sin(x) > cos(x)$ *y* $x \in [-6,6]$.

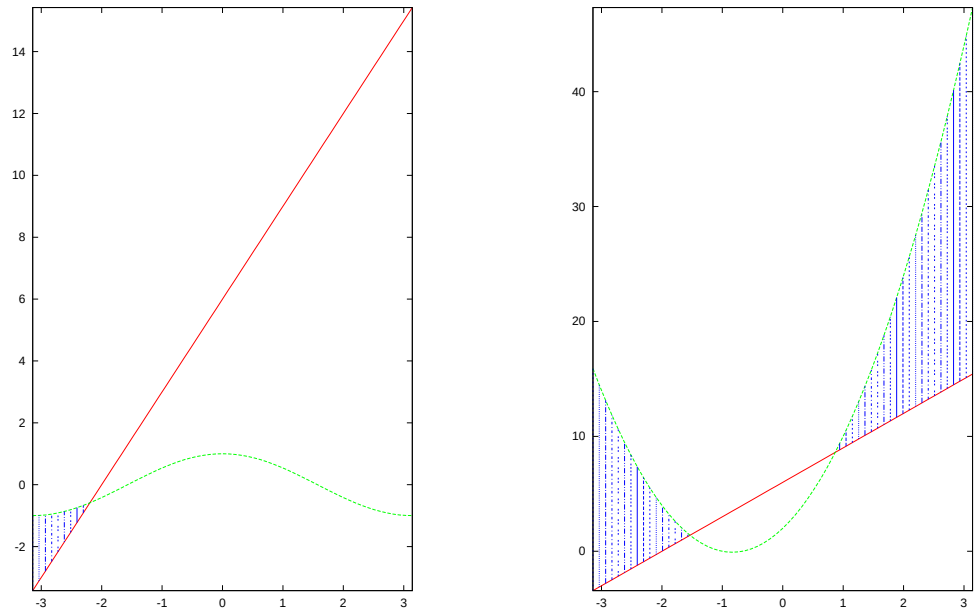

Figura 5: A la izquierda el gráfico correspondiente al ejemplo 5 y a la derecha el correspondiente al ejemplo 6.

Las funciones *sin* y *cos* son funciones de la biblioteca, pero como se muestra en los siguientes ejemplos también se pueden usar funciones propia.

Ejemplo 5 *Sombrear la región encerrada entre por los gráficos de* $f(x) = 3*x+6$ *y* $g(x) = cos(x)$ *donde* $3*x+6 < cos(x)$ *y* $x \in [-\pi, \pi]$.

pi es una función de biblioteca que nos devuelve una aproximación del valor de π.

Ejemplo 6 *Sombrear la región encerrada entre por los gráficos de* $f(x) = 3*x+6$ *y* $g(x) = 3*x^2 + 5*x+2$ *donde* $3*x+6 < 3*x^2+5*x+2$ *y* $x \in [-\pi, \pi]$.

A continuación se muestran las definiciones de las funciones utilizadas en los ejemplos precentes.

recta

```
function f=recta(x)
  f=3*x+6;
endfunction
```