Señales y sistemas

Ejemplo de ecuación:

$$\int_{-\infty}^{\infty} e^{-at} dt$$

$x = sin(\omega t)$

Señales continuas y discretas

Se tomará la covención:

Señal continua := x(t), t variable de tiempo

Las señales discretas := x[n]

Function Handles vs Simbólicas

```
%Crear una variable simbólica
syms t

% Función simbólica:
y_symb = cos(t)
```

y_symb = $\cos(t)$

```

%Function handle
%sintaxis: variable = @(parámetro) función
y_fh = @(t) cos(t);
```

Pero la diferencia más importante es que "function handle" actúa como

una "Lambda de Python" y podemos evaluar facilmente:

```
y_fh([5 8 10])
```

```
ans = 1×3
    0.2837   -0.1455   -0.8391
```

Pero con las simbólicas no se puede hacer asi y se tiene que utilizar:

```
% No se puede utilizar sólo la función subs;
% Se tiene que utilizar en conjunto con la función
% "double":
ans_subs_y_symb = subs(y_symb,t,[5 8 10])
```

ans_subs_y_symb = $\begin{pmatrix} \cos(5) & \cos(8) & \cos(10) \end{pmatrix}$

```
double(ans_subs_y_symb)
```

```
ans = 1×3
    0.2837   -0.1455   -0.8391
```

Se puede hacer álgebra con variables simbólicas:

```
y_3 = y_symb + sin(t) + exp(-t)
```

y_3 = $e^{-t} + \cos(t) + \sin(t)$

```
% Diferenciar
diff(y_3)
```

ans = $\cos(t) - e^{-t} - \sin(t)$

```
% Integrar. Nuevamente, como con la substitución,
% tenemos que utilzar la función "Double" para evaluarla:

ans_int = int(y_3,[0 1])
```

ans_int = $\sin(1) - e^{-1} - \cos(1) + 2$

```
double(ans_int)
```

```
ans = 1.9333
```

```
int(y_3,t)
```

ans =

$$-e^{-t} - \sqrt{2}\,\cos\left(t + \frac{\pi}{4}\right)$$

No se pueden hacer operaciones con Function Handles:

```
% y_fh + y_fh Error!
```

Se pueden convertir las funciones simbólicas en function handles:

```
y_3_fh = matlabFunction(y_3)
```

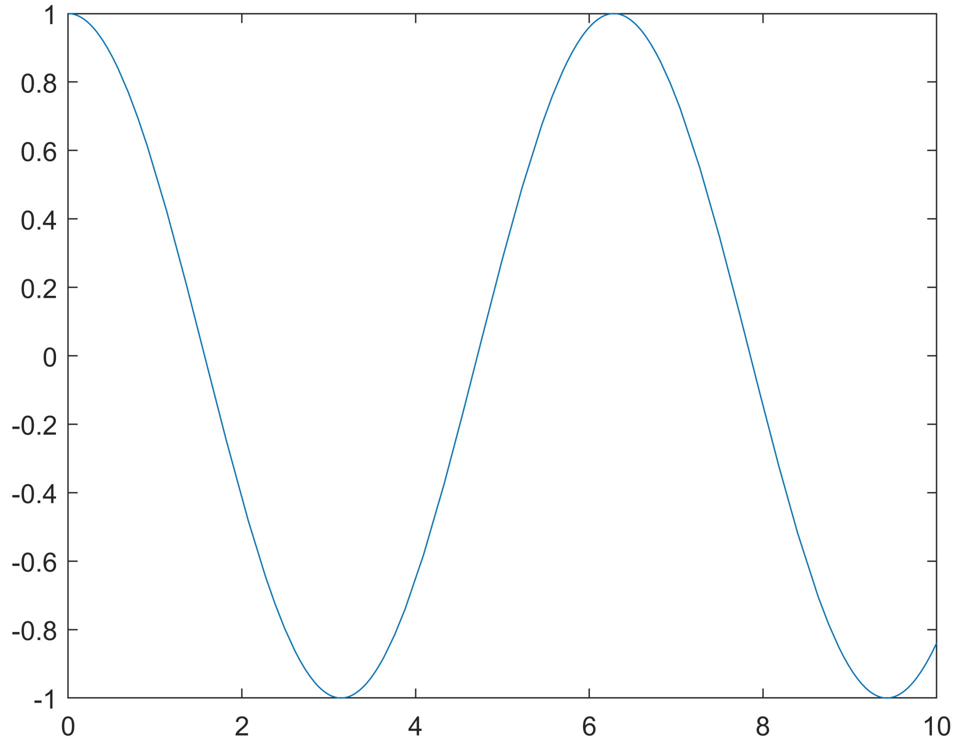
```
y_3_fh = function_handle with value:
    @(t)exp(-t)+cos(t)+sin(t)
```

```
y_3_fh([5 8 10])
```

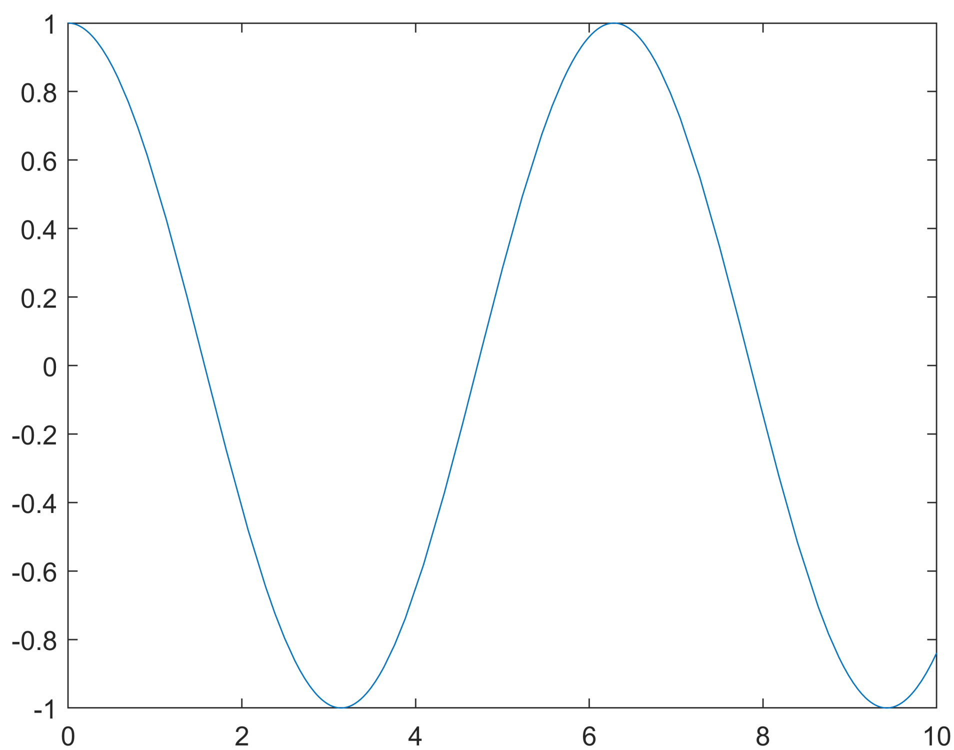
```
ans = 1×3
   -0.6685    0.8442   -1.3830
```

Graficas funciones continuas vs discretas

```
% Estas son las gráficas de las funciones del inicio del Livescript
%fplot es para funciones simbólicas

figure
fplot(y_symb,[0,10])
```




```
figure
fplot(y_fh,[0,10])
```

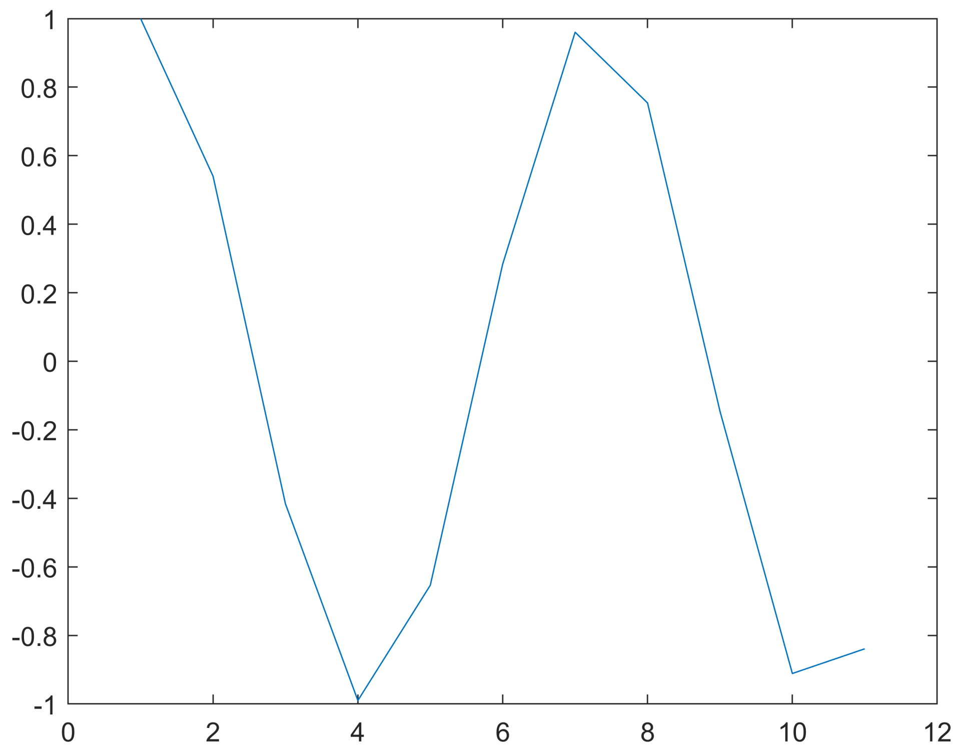


```
%pasa lo mismo
```

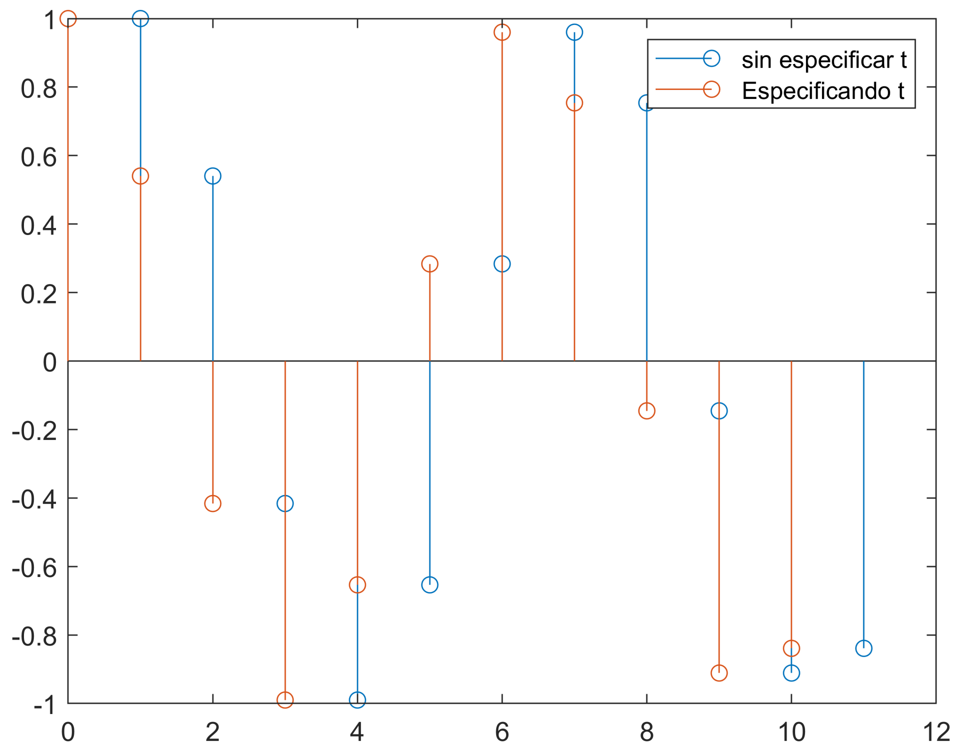
```
%el ";" suprime la salida
% generamos puntos discretos
t_disc = 0:10;
y_disc = y_fh(t_disc)
```

```
y_disc = 1×11
    1.0000    0.5403   -0.4161   -0.9900   -0.6536    0.2837    0.9602    0.7539 ···
```

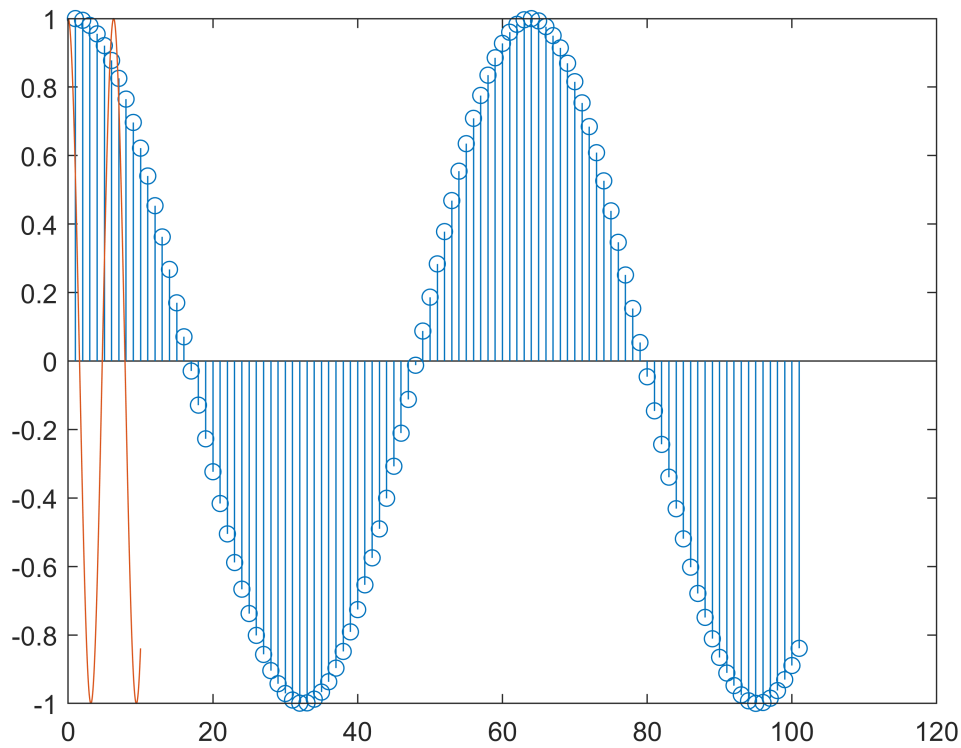
```

figure
plot(y_disc)
```



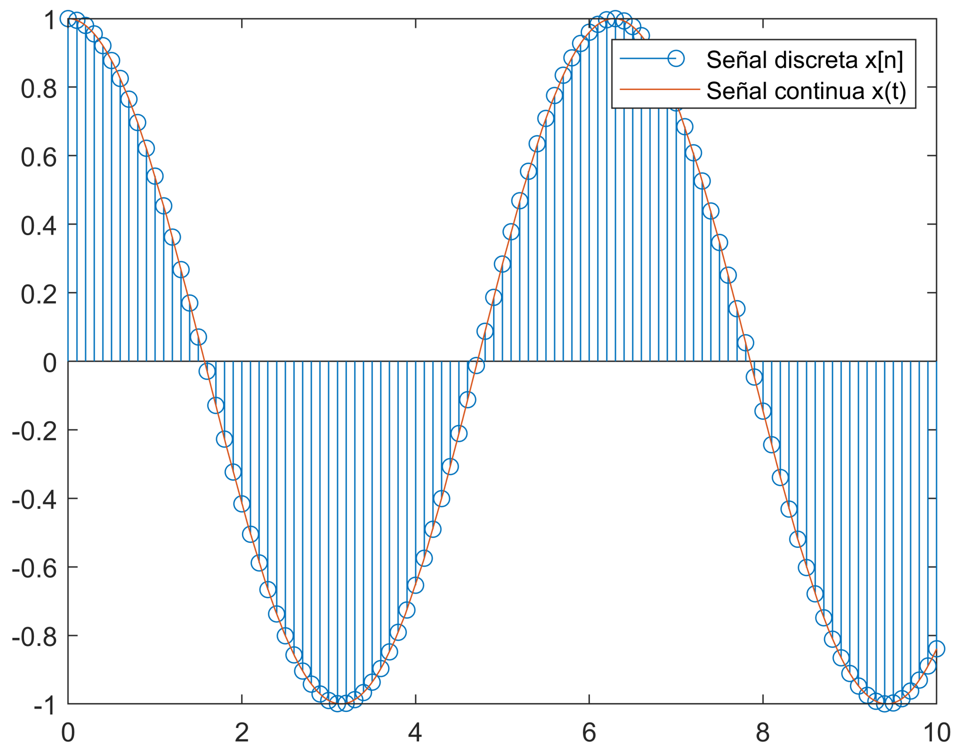

```
%stem es para muestras o vectores (las líneas verticales. stem=tallo)

% Mostramos que, si no especificamos dónde inicia el eje x,
% nuestro plot iniciará desde 1:
figure
stem(y_disc)
hold on
stem(t_disc,y_disc)
hold off
legend("sin especificar t","Especificando t")
```




```
%generamos puntos discretos más densos

t_disc = 0:0.1:10;
y_disc = y_fh(t_disc);

%Volvemos a ver qué pasa si no specificamos el valor del eje x
figure
stem(y_disc)
hold on
fplot(y_fh,[0,10])
hold off
```

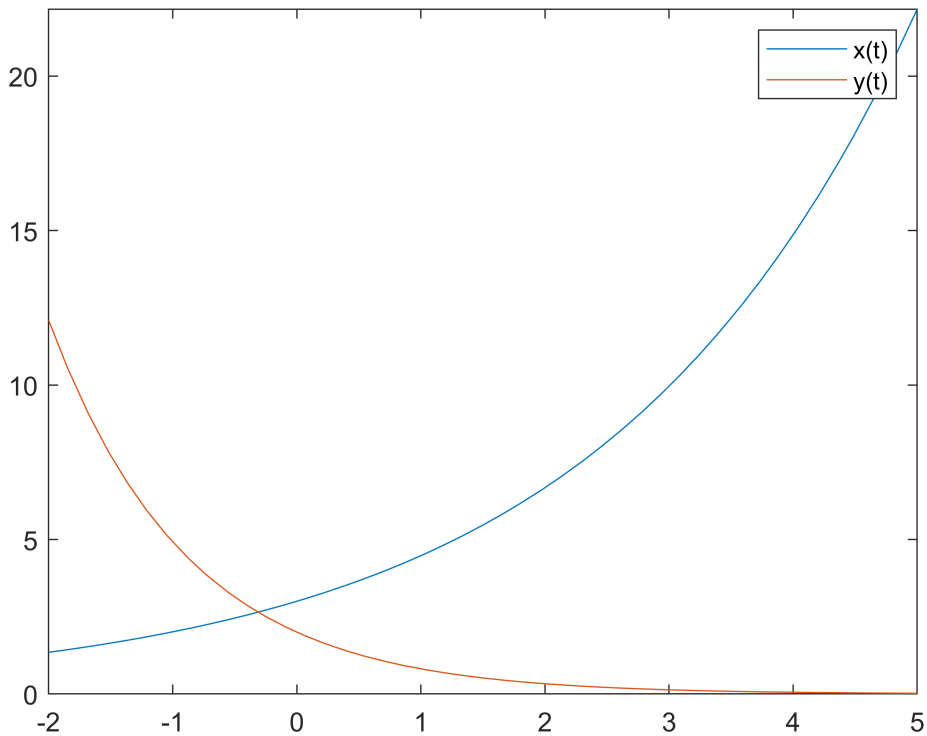



```
%Si ahora especificamos el eje x:

figure
stem(t_disc,y_disc)
hold on
fplot(y_fh,[0,10])
xlim([0,10])
legend("Señal discreta x[n]","Señal continua x(t)")
hold off
```

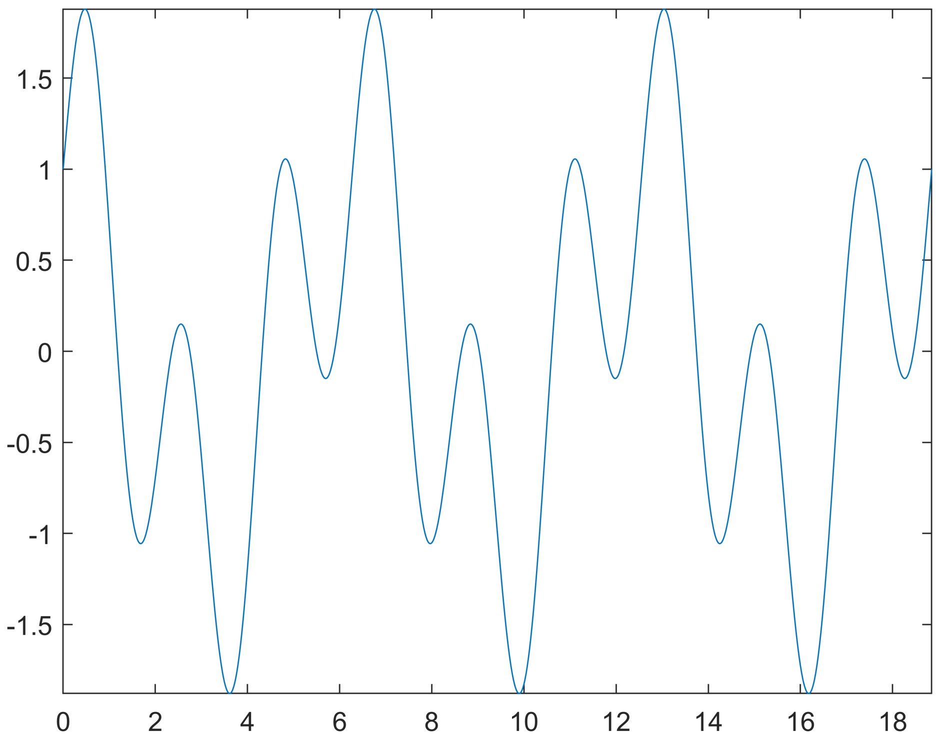


Formas de onda básicas (ejercicios):

```
%utilizo '.*' porque es la forma
% de hacer multiplicación directa pues Matlab
% trabaja con matrices y '*' es hacer el producto punto de
% matrices

%Ejercicio 1:

x_fh1 = @(t) 3.*exp(0.4.*t);
y_fh1 = @(t) 2.*exp(-0.9.*t);

%Plotemos las funciones anteriores:
figure
fplot(x_fh1,[-2,5])
hold on
fplot(y_fh1,[-2,5])
legend("x(t)","y(t)")
hold off
```

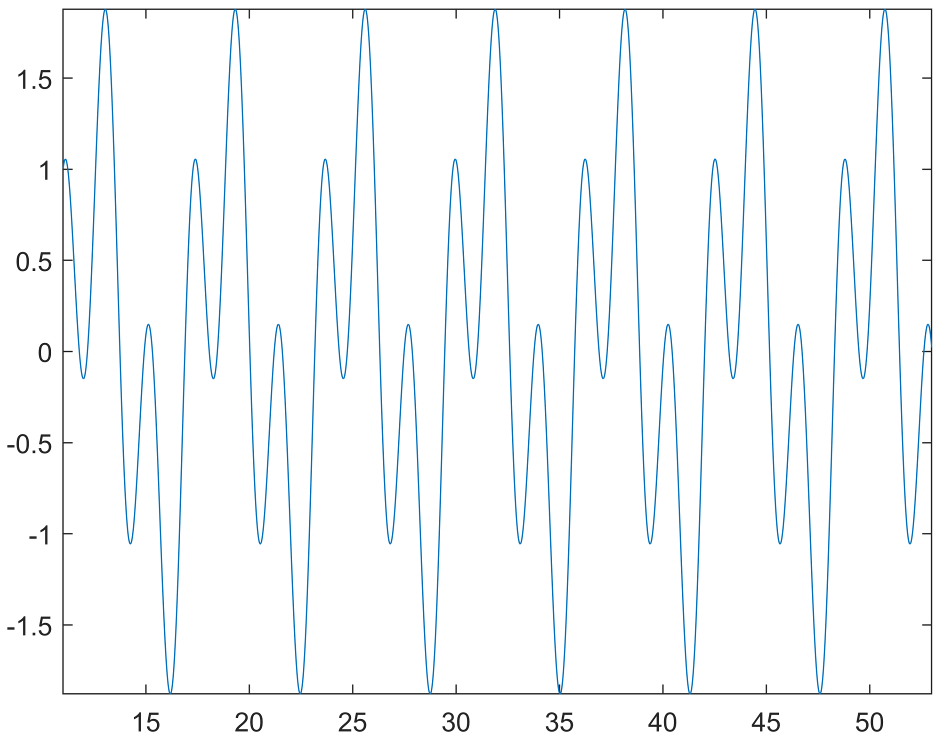



```
%Ejercicio 2:
x_fh2 = @(t) cos(t) + sin(3.*t);

figure
fplot(x_fh2,[0,6*pi])
```

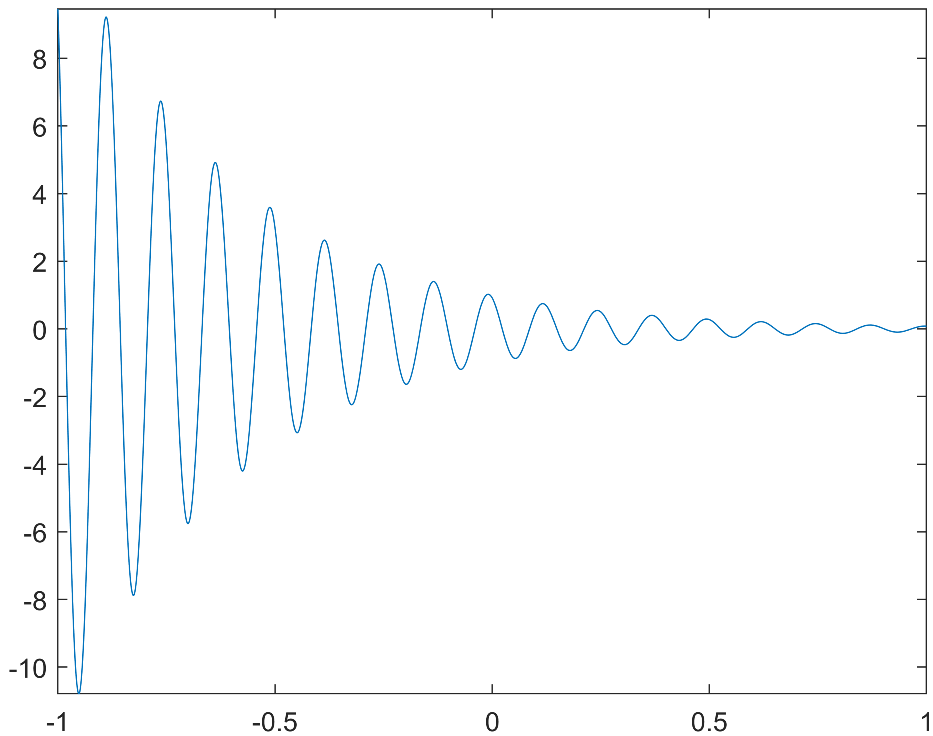


Usar sliders

```
% Hacemos sliders para los coeficientes de la función
a = 1.4;
b = 2.6;

%También un slider para los límites del eje x:

t0 = 11;
tf = 53;

x_fh3 = @(t) cos(a.*t) + sin(b.*t);

figure
fplot(x_fh2,[t0,tf])
```




```
% Ejemplo 3

% Hacemos sliders para los coeficientes de la función
theta = -31;
omega_0 = 50;
C = 1;
r = -2.5;

x_fh4 = @(t) C.*exp(r.*t).*cos(omega_0.*t + theta);

figure
fplot(x_fh4,[-1,1])
```


8
6
4
2
0
-2
-4
-6
-8
-10
-1
-0.5
0
0.5
1