Modelo para los ejercicios de la practica 3

En este documento se recogen alguna de las soluciones de los ejercicios propuestos en la sesion III.

La salida que devuelve Matlab está en gris claro.

`Comandos utilizados en esta practica`

`inline, vectorize, plot, limit`

Nota Uno de los mayores errores radica en el manejo de los paréntesis cuando se introducen expresiones. Con un poco de práctica se soluciona rápido. De todas formas conviene tener en cuenta:

a) No pongas paréntesis no necesarios salvo que ayude, y mucho, a la legibilidad.

b) Cada vez que abras un paréntesis, ciérralo y ponte en medio. Te ayudará al menos para tener bien claro que se abren los mismos paréntesis que se cierran

c) Utiliza la instrucción `pretty` (si tienes la toolbox de simbólico a mano, claro) para comprbar que la expresión es la correcta.

d) Y por supuesto, aprende bien el orden de las operaciones `+`, `-`, `*`, `./`, `^`; esto no es exclusivo de Matlab sino que forma parte de cualquier lenguaje de programación.

Dicho esto, arrancamos con esta guía de *problemas resueltos*

Contents

Ejercicio 1.1 Dibujar algunas funciones

Función 1.1(a)

$x\mathrm{sen}(x)$

Definimos la función (recuerda que `sin` es la función seno)

```
f=vectorize(inline('x*sin(x)','x'))
```

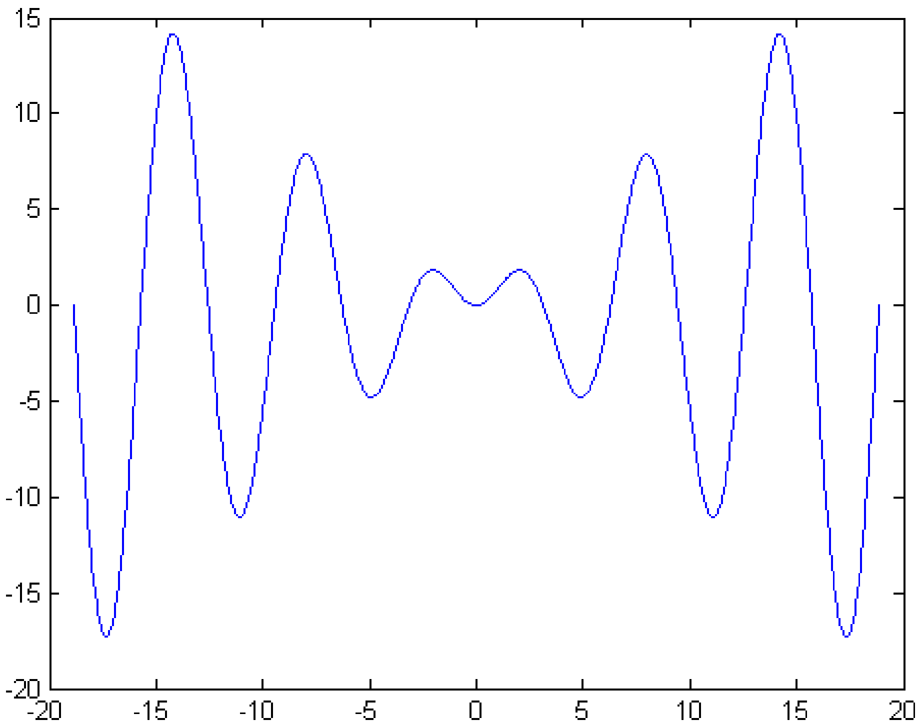
```
f =

     Inline function:
     f(x) = x.*sin(x)
```

Elegimos intervalo de dibujo, (-6*pi,6*pi), y número de puntos 1000

```
t=linspace(-6*pi,6*pi,1000);
```

Dibujamos

```
plot(t,f(t))
```



Observa que utilizamos `x` para definir la función y posteriormente como variable simbólica, mientras que `t` es la variable usada para dibujar. No es relevamente. La instrucciones siguiente daría el mismo dibujo

```
mivariable_t= linspace(-6*pi,6*pi,1000); plot(mivariable_t,f(mivariable_t))
```

Función 1.1(c)

$x\sqrt{\cos(x)}$

Introducimos la función

```
f3=vectorize(inline('x*sqrt(cos(x))','x'))
```

```
f3 =

     Inline function:
     f3(x) = x.*sqrt(cos(x))
```

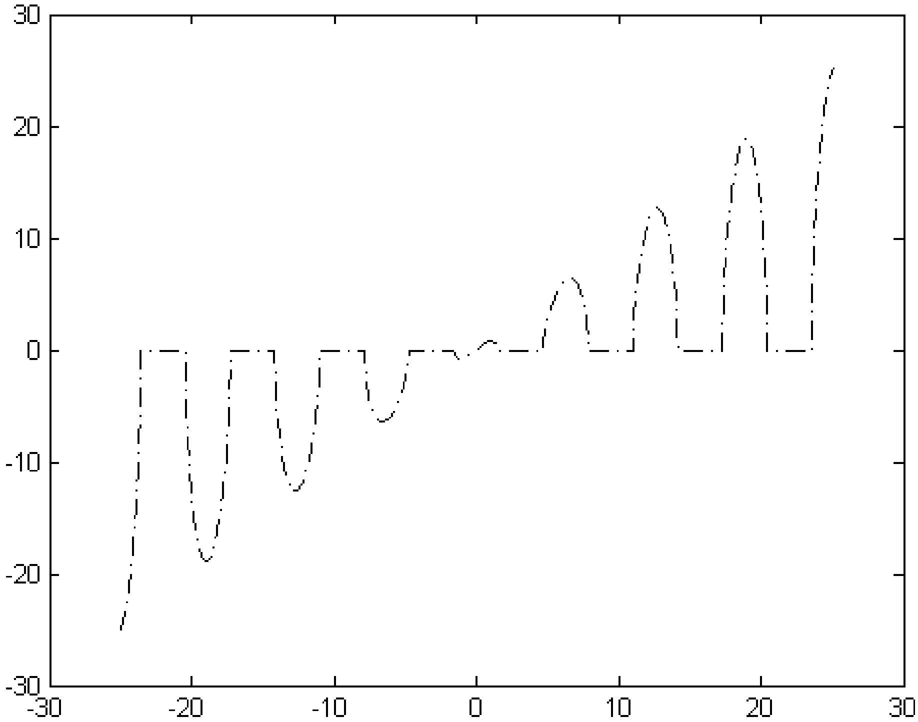
Elegimos el intervalo de dibujo, (-8*pi,8*pi), y número de puntos igual a 1000

```
t=linspace(-8*pi,8*pi,1000);
```

Dibujamos y *tuneamos* un poco la grafica

```
plot(t,f3(t),'k-.')
```

```
Warning: Imaginary parts of complex X and/or Y arguments ignored
```



Observa que se devuelve un aviso. Esencialmente, se informa de que para algunos valores, la función es compleja (donde?) y que en esos sitios no se dibuja

Función 1.1.(d)

$$\frac{\cos(x)}{(1-x^2/2)}$$

```
f4=vectorize(inline('cos(x)/(1-x^2/2)','x'))
```

```
f4 =

     Inline function:
     f4(x) = cos(x)./(1-x.^2./2)
```

para asegurarnos que la función está bien escrita, podemos utilizar el comando `pretty` (requiere que la toolbox de simbólico esté instalada)

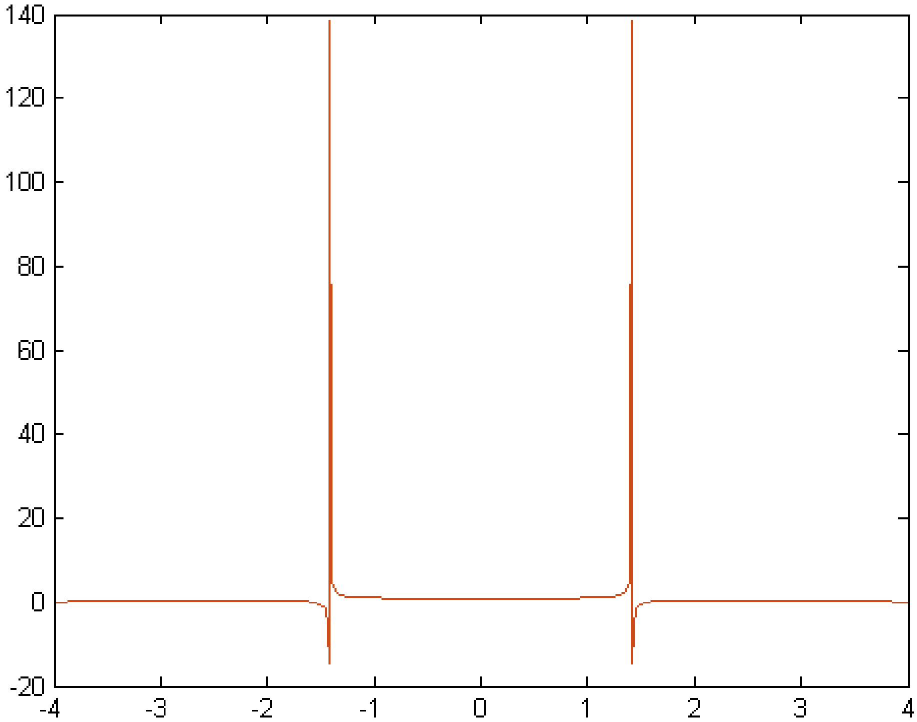
```
syms x
pretty(f4(x))
```

```
    cos(x)
  ----------
           2
```

```
1 - 1/2 x
```

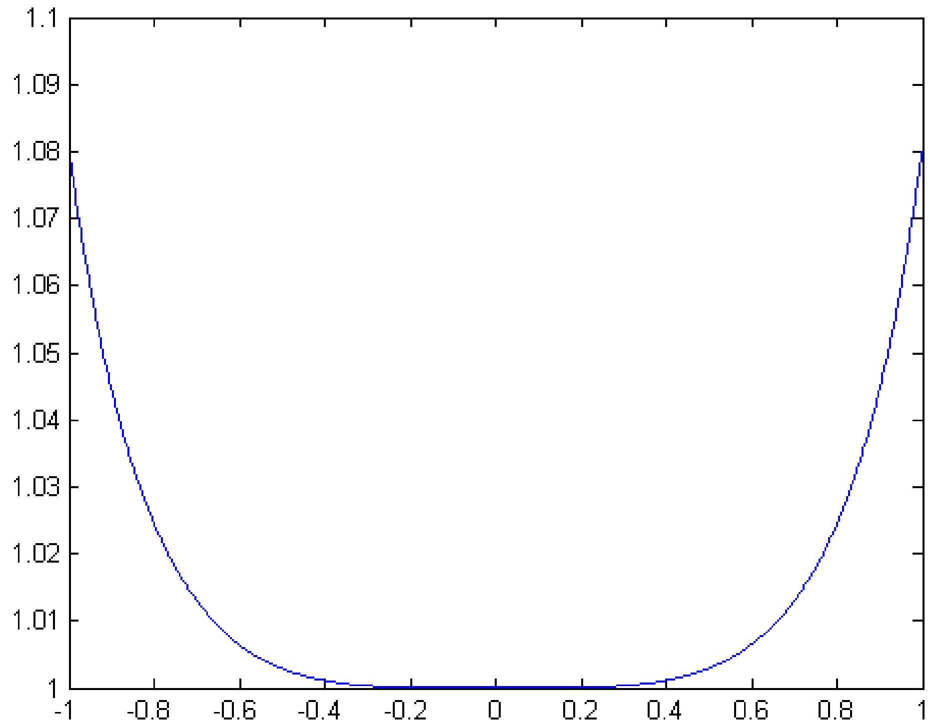
es correcto. Dibujamos entre -4,4

```
t=linspace(-4,4,1000);
plot(t,f4(t),'linewidth',1.25,'color',[0.8,0.3,0.1])
```



Vemos que hay asíntotas verticales. Redibujamos cerca del cero, por ejemplo, entre (-1,1)

```
t=linspace(-1,1,1000);
plot(t,f4(t))
```



Ejercicio 1.2 Dibujar simultaneamente algunas funciones

El objetivo del presente ejercicio es que visualicéis juntas algunas funciones y sus "equivalencias", cerca del cero. En algunos casos habrá¡ que hacer "zooms" fuertes para ver mínimamente la diferencia entre ambas

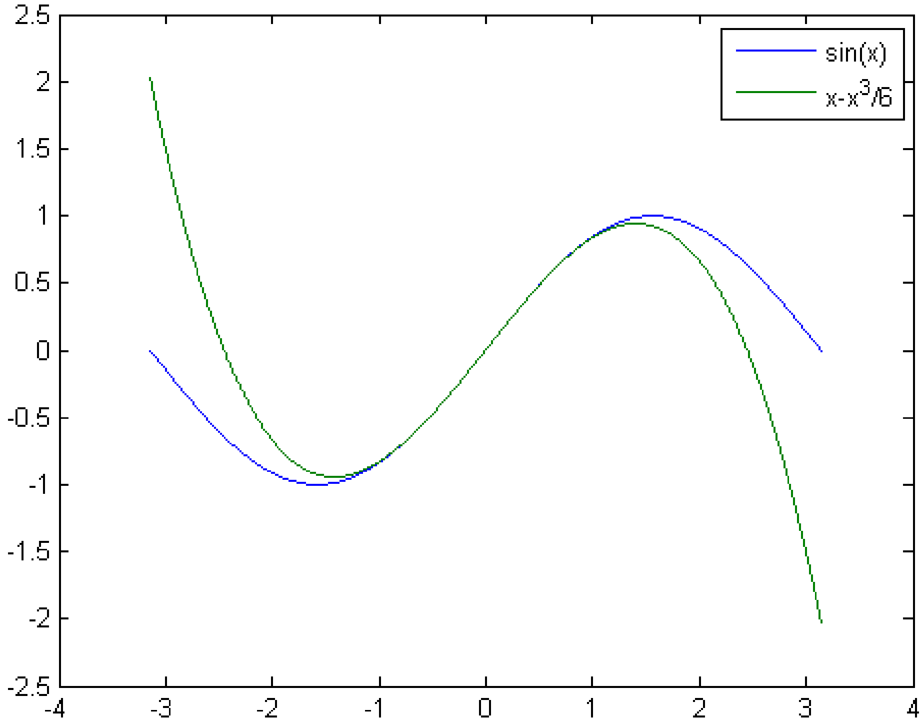
Función c

```
fc=vectorize(inline('sin(x)','x'))
fc2=vectorize(inline('x-x^3/6','x'))
```

```
fc =

     Inline function:
     fc(x) = sin(x)


fc2 =

     Inline function:
     fc2(x) = x-x.^3./6
```

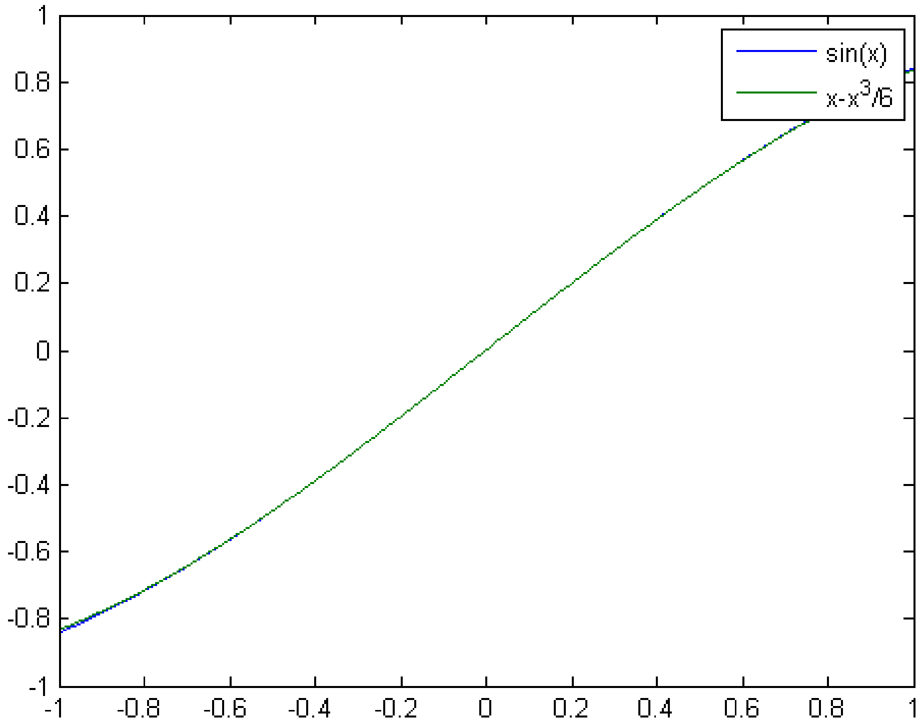
Tomamos 5000 puntos en [-\pi,\pi]

```
t=linspace(-pi,pi,5000);
plot(t,fc(t),t,fc2(t))
legend('sin(x)','x-x^3/6')
```



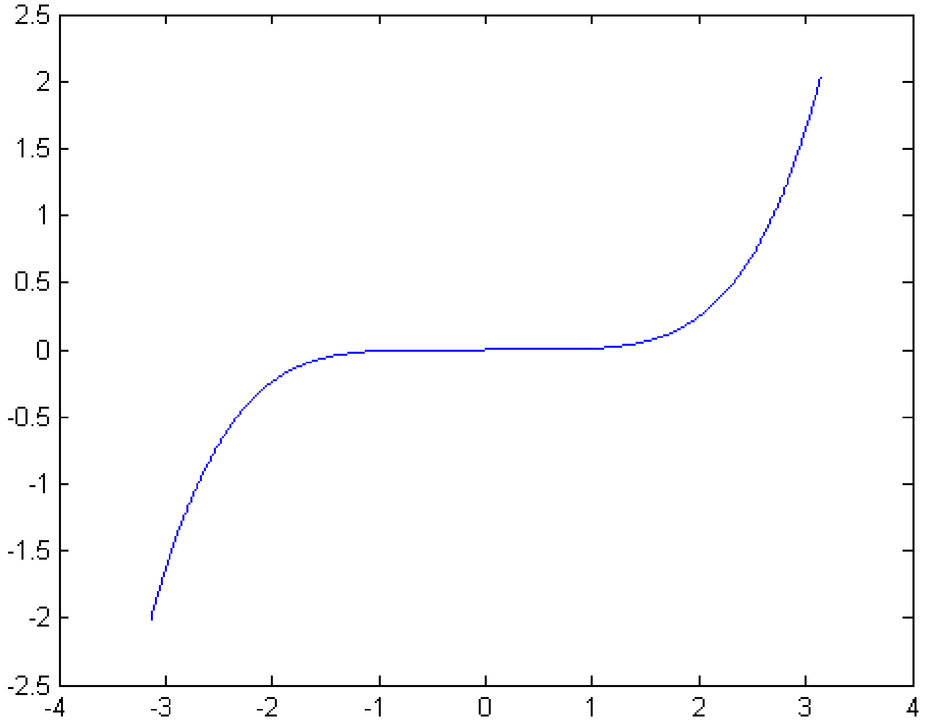
No vemos mucha semejanza... Nos acercamos al cero, bien dando otros valores a t y redibujando, o simplemente con xlim

```
xlim([-1,1]) % x en [-1,1]
```



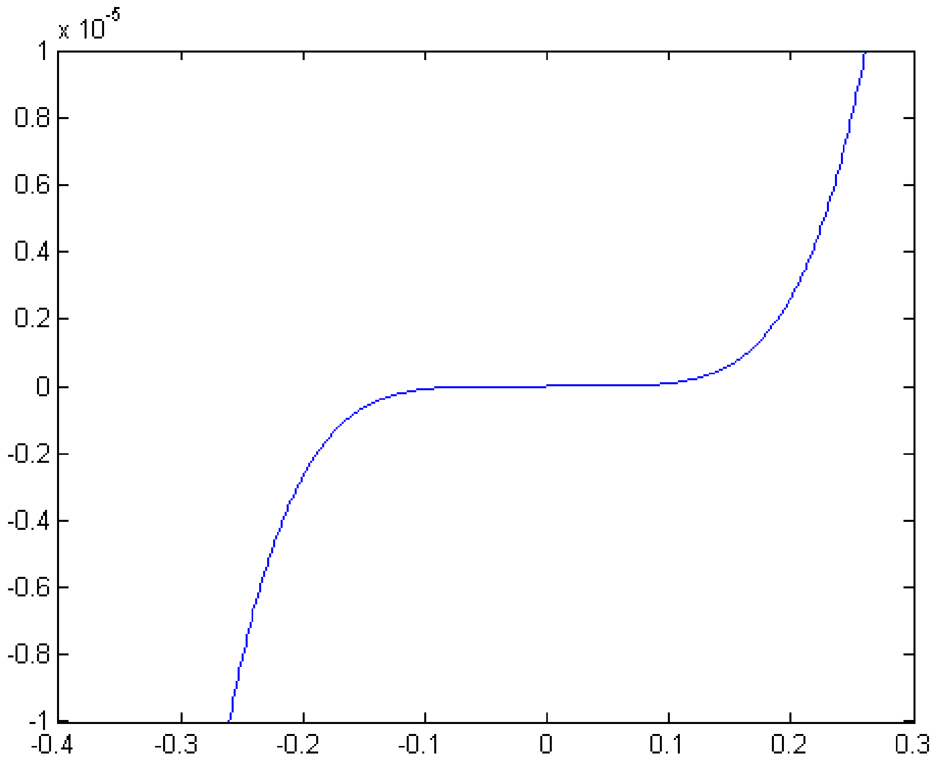
Otra forma de ver su parecido es dibujar la función diferencia:

```
plot(t,fc(t)-fc2(t))
```



y ahora para valores muy cercanos a cero

```
ylim([-1e-5,1e-5])
```



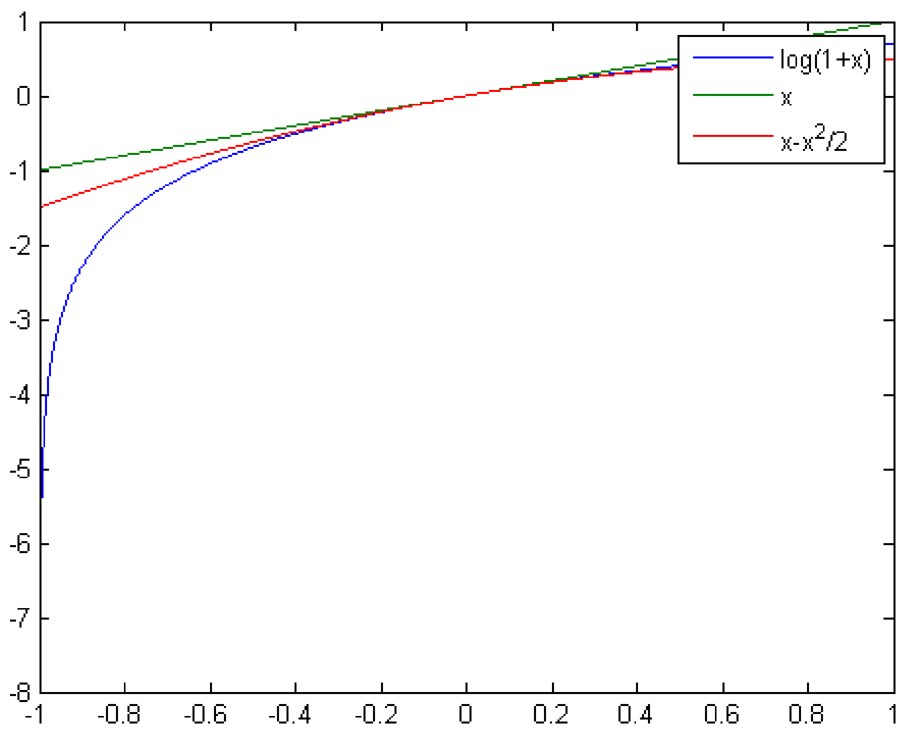
Función d

```
fd=vectorize(inline('log(1+x)','x'))
fd2=vectorize(inline('x','x'))
fd3=vectorize(inline('x-x^2/2','x'))
```

```
fd =

     Inline function:
     fd(x) = log(1+x)


fd2 =

     Inline function:
     fd2(x) = x


fd3 =

     Inline function:
     fd3(x) = x-x.^2./2
```

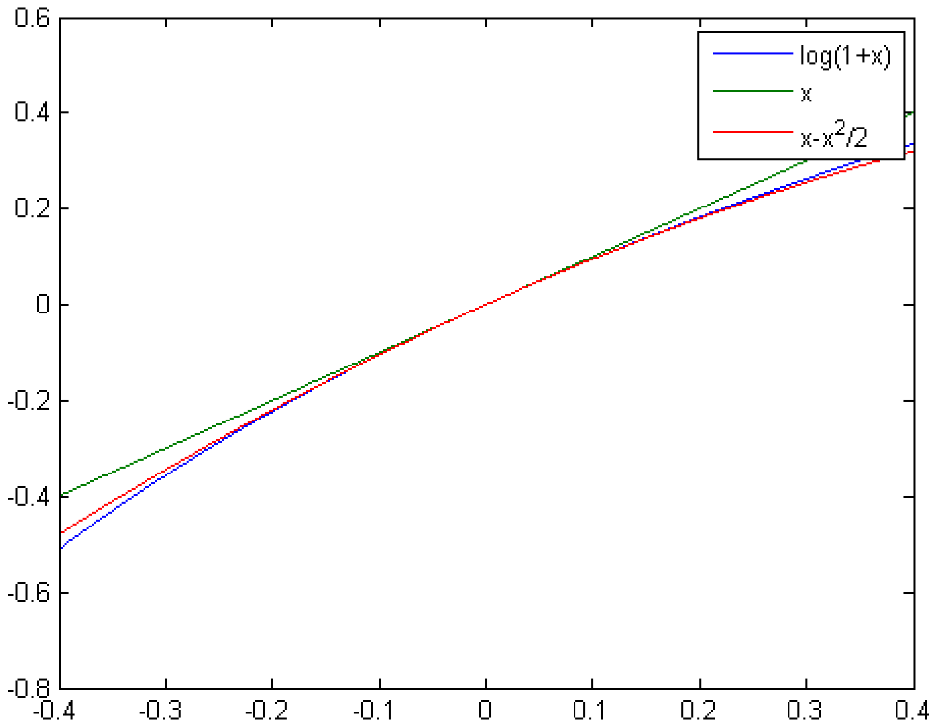
Tomamos 5000 puntos en [-1,1]. Observa que cuando x=-1, hay una asíntota (por que?)

```
t=linspace(-1,1,5000);
plot(t,fd(t),t,fd2(t),t,fd3(t))
legend('log(1+x)','x','x-x^2/2')
```



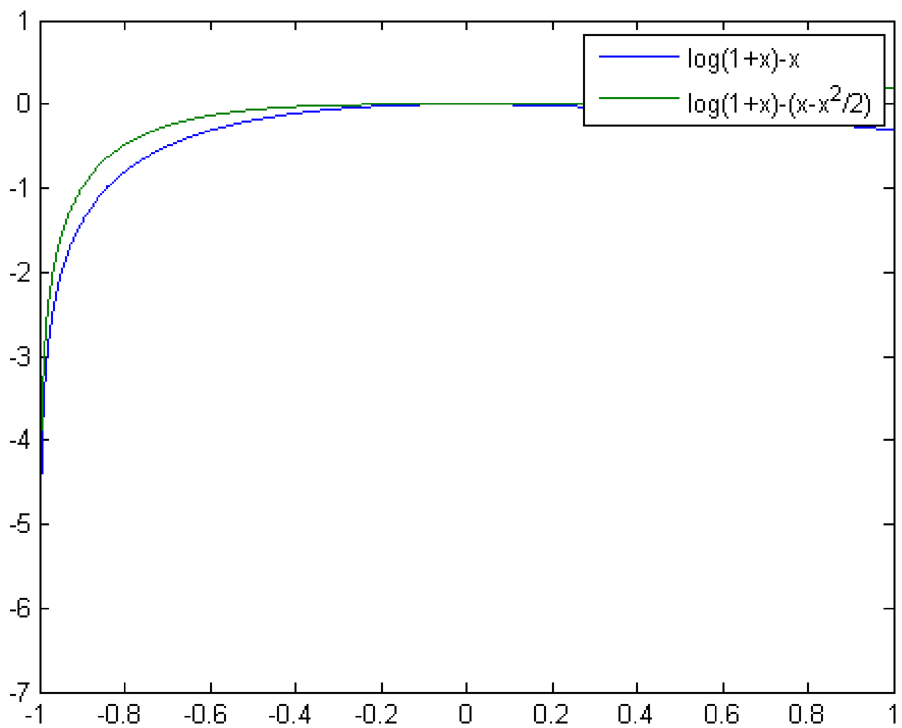
De nuevo debemos acercanos para ver que son muy parecidas cerca (pero solo cerca) del cero

```
xlim([-0.4,0.4]) % x en [-1,1]
```



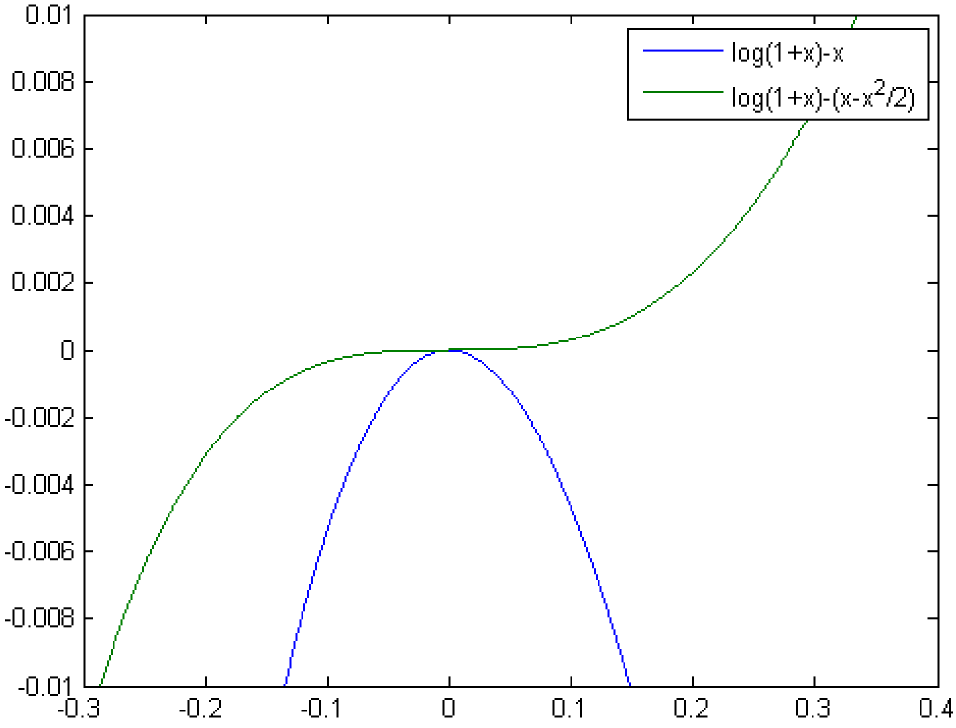
Podemos dibujar otra la función diferencia:

```
plot(t,fd(t)-fd2(t),t,fd(t)-fd3(t))
legend('log(1+x)-x', 'log(1+x)-(x-x^2/2)')
```



y ahora cerca de 0

```
ylim([-1e-2,1e-2])
```



Problema 1.4

Cálculo de limites

Función 1.4a

```
f=vectorize(inline(' ((3*x-1)/(x+1))^((x+1)/(x-1))','x'))
```

```
f =

     Inline function:
     f(x) = ((3.*x-1)./(x+1)).^((x+1)./(x-1))
```

Vemos f para ver si está bien introducida

```
syms x
pretty(f(x))
```

```
                         /1 + x\
                         |-----|
                         \x - 1/
               /3 x - 1\
               |-------|
               \ 1 + x /
```

Calculamos el limite

```
limit(f(x),x,1)
```

```
ans =

exp(2)
```

Función 1.4b

```
f=vectorize(inline(' (1/(x-3))^((x^2-x-6)/(2*x^2-4*x-6))','x'))
```

```
f =

     Inline function:
     f(x) = (1./(x-3)).^((x.^2-x-6)./(2.*x.^2-4.*x-6))
```

Vemos f para ver si está bien introducida

```
syms x
pretty(f(x))
```

```
                                   /    2           \
                                   |   x  - x - 6   |
                                   |----------------|
                                   |    2           |
                                   \2 x  - 4 x - 6/
                            /  1  \
                            |-----|
                            \x - 3/
```

Calculamos el límite

```
limit(f(x),x,3)
```

```
ans =

limit((1/(x-3))^((x^2-x-6)/(2*x^2-4*x-6)),x = 3)
```

Devuelve `Nan`. el límite no existe. Hacemos límite por direcciones

```
limit(f(x),x,3,'right')
limit(f(x),x,3,'left')
```

```
ans =

Inf
```

```
ans =

1/(1/2*2^(1/2)-1/2*i*2^(1/2))^(5/2)*Inf
```

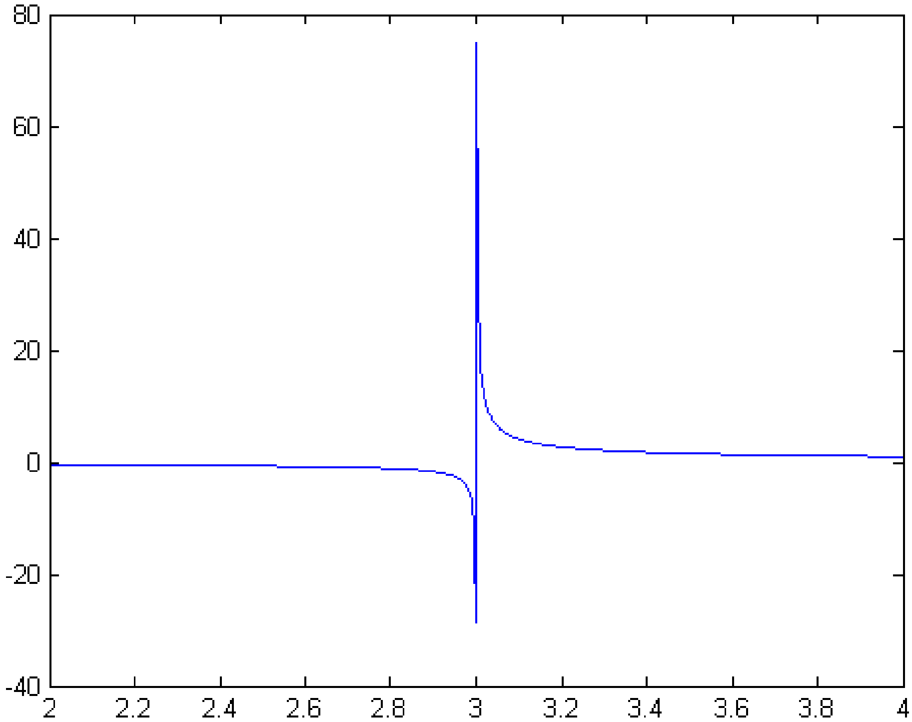
Lo que sucede es que la función

$$\left(\frac{1}{x-3}\right)^{\frac{x^2-x-6}{2x^2-4x-6}}$$

solo existe para x>3:

```
t=linspace(2,4,1000);
plot(t,f(t))

% Si observamos está avisando de que tiene valores complejos, que
% corresponde a los puntos t<3 (Matlab dibuja la parte real)
```

```
Warning: Imaginary parts of complex X and/or Y arguments ignored
```



Función 1.4d

```
f=vectorize(inline(' log(1+x)/(cos(exp(x^2)-1)-1)','x'))
```

```
f =

     Inline function:
     f(x) = log(1+x)./(cos(exp(x.^2)-1)-1)
```

Vemos f para ver si está bien introducida

```
syms x
pretty(f(x))
```

```
      log(1 + x)
--------------------
          2
cos(exp(x ) - 1) - 1
```

Calculamos el límite

```
limit(f(x),x,0)
```

```
ans =

NaN
```

Límites laterales

```
limit(f(x),x,0,'right')
limit(f(x),x,0,'left')
```

```
ans =

-Inf


ans =

Inf
```

asíntota en x=0

Función 1.4g

```
f=vectorize(inline(' sqrt(x-sqrt(x))-sqrt(x)','x'))
```

```
f =

     Inline function:
     f(x) = sqrt(x-sqrt(x))-sqrt(x)
```

Vemos f para ver si está bien introducida

```
syms x
pretty(f(x))
```

```
            1/2 1/2     1/2
      (x - x    )    - x
```

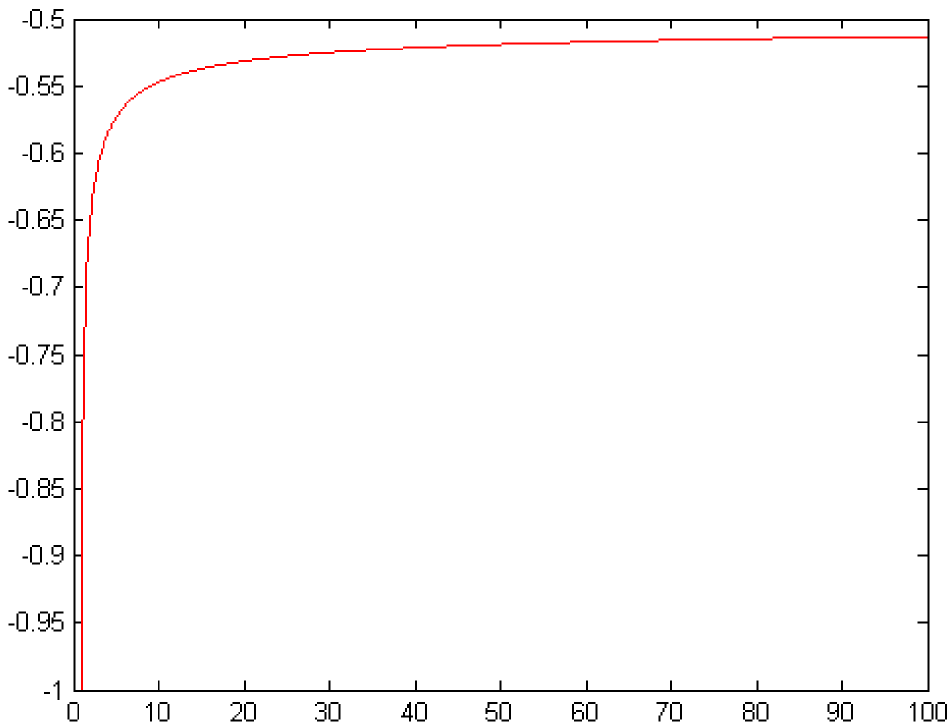
Calculamos el límite

```
limit(f(x),x,inf)
```

```
ans =

-1/2
```

Un dibujo para ver esto

```
t=linspace(1,100,1000);
plot(t,f(t),'r-')
```



Función 1.4j

```
f=vectorize(inline('(x*cos(sqrt(abs(x)))-x)/(sin(x^2))','x'))
```

```
f =

     Inline function:
     f(x) = (x.*cos(sqrt(abs(x)))-x)./(sin(x.^2))
```

Comprobamos si f está bien introducida

```
syms x
pretty(f(x))
```

```
                 1/2
  x cos(| x |   ) - x
  -------------------
             2
        sin(x )
```

Calculamos el límite

```
limit(f(x),x,0)
```

```
ans =

NaN
```

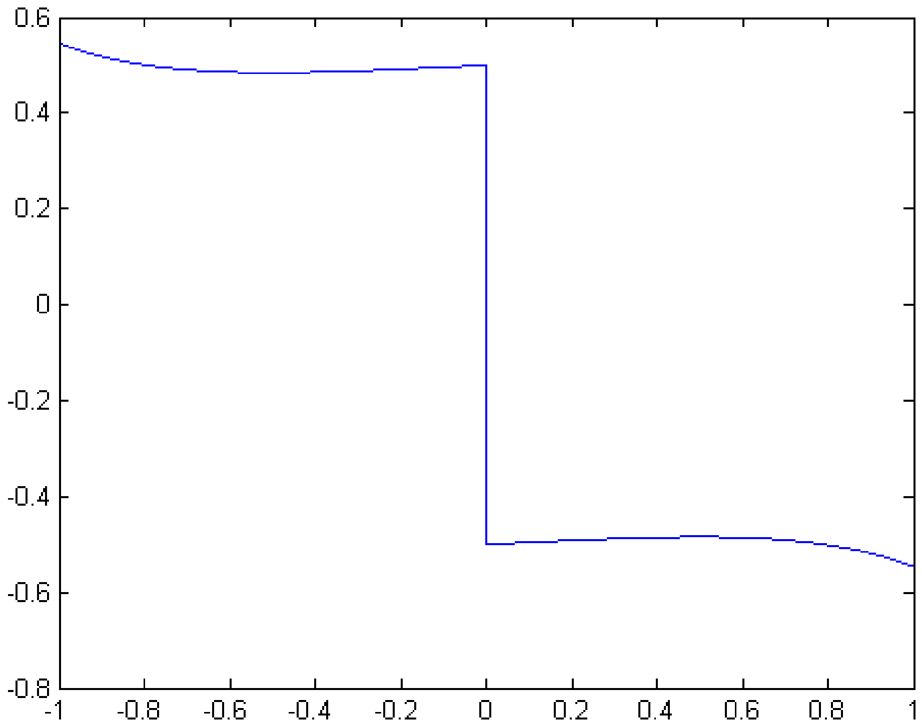
No existe. Procedemos a hacer el límite por direcciones:

```
limit(f(x),x,0,'left')
limit(f(x),x,0,'right')
```

```
ans =

1/2


ans =

-1/2
```

Un dibujo para ver esto

```
t=linspace(-1,1,1000);
plot(t,f(t))
```



Problema 1.5 Cálculo de asíntotas

Problema 1.5c

$$\frac{x^3-2x^2+1}{x^2+4x-5}$$

Introducimos la función

```
f=vectorize(inline('(x^3-2*x^2+1)/(x^2+4*x-5)','x'))
```

```
f =

     Inline function:
     f(x) = (x.^3-2.*x.^2+1)./(x.^2+4.*x-5)
```

Comprobamos que está bien introducida

```
syms x
pretty(f(x))
```

```
 3      2
x  - 2 x  + 1
-------------
 2
x  + 4 x - 5
```

Todo correcto.

Pasamos a buscar asíntotas horizontales: límite en el infinito

```
limit(f(x),x,inf)
limit(f(x),x,-inf)
```

```
ans =

Inf


ans =

-Inf
```

No hay asíntotas horizontales.

asíntotas oblicuas en infinito

```
limit(f(x)/x,x,inf)
```

```
ans =

1
```

Parece que si. Recogemos el valor del límite utilizando ans o bien tecleando

`m=limit(f(x)/x,x,infty)`

al principio

```
m=ans;
b=limit(f(x)-m*x,x,inf)
```

```
b =

-6
```

por tanto, y=x+6 es una asíntota en el infinito. Podemos proceder igual en menos infinito.

```
m2=limit(f(x)/x,x,-inf)
b2=limit(f(x)-m2*x,x,-inf)
```

```
m2 =

1


b2 =
```

```
-6
```

Asítotas verticales. Estas requieren encontrar los puntos donde la función tiende a mas o menos infinito.

Vamos alla. Utilizamos una nueva función, `solve`, que resuelve ecuaciones

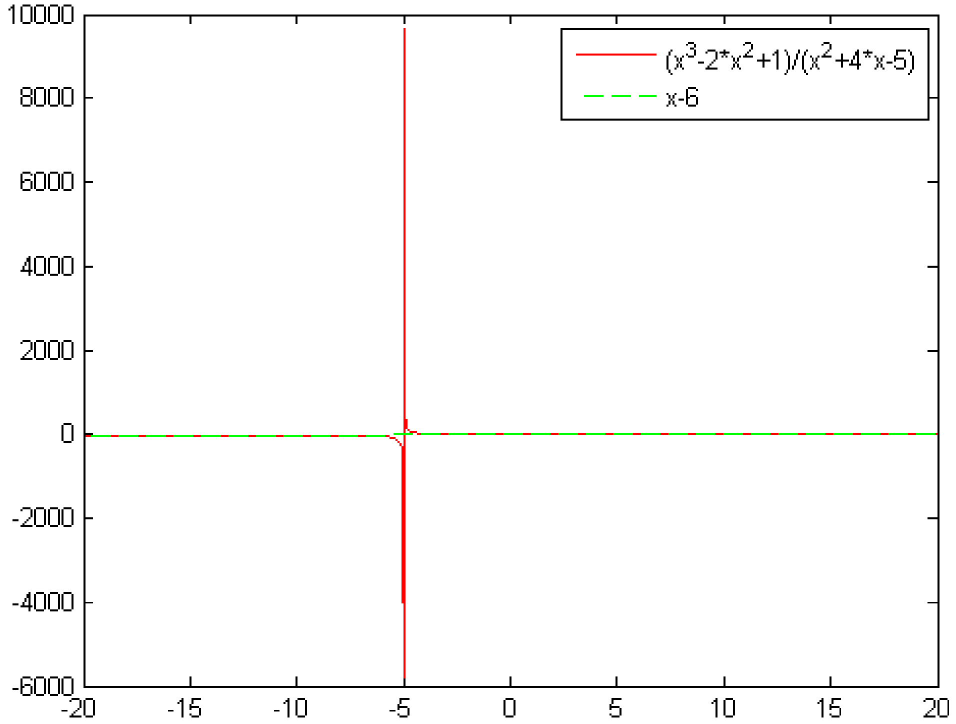
```
solve(x^2+4*x-5,'x')
%
% Dos posibles puntos, -5 y 1
limit(f(x),x,-5,'right')
limit(f(x),x,-5,'left')
limit(f(x),x,1,'right')
limit(f(x),x,1,'left')
```

```
ans =

  1
 -5


ans =

Inf


ans =

-Inf


ans =

-1/6


ans =

-1/6
```

Encontramos por tanto una única asíntota en -5 (en -1 hay límite). Finalmente podemos dibujar función y asíntotas oblicuas (la vertical se ve más claramente)

```
asintota=vectorize(inline('x-6','x'))
t=linspace(-20,20,5000);
plot(t,f(t),'r',t,asintota(t),'g--')
legend(char(f(x)),char(asintota(x)))
```

```
asintota =

     Inline function:
     asintota(x) = x-6
```



`char` es una función que transforma todo a "carácteres" (*strings*). Se puede utilizar, como en este caso, para no tener que teclear diréctamente el nombre de la función.

En cualquier caso, no vemos literalmente nada porque Matlab trata de captar toda la gráfica de la función y da valores desproporcionados. "Cortamos en y"

```
ylim([-40,40])
```