Estadística y Métodos Matemáticos para la Investigación: Métodos Numéricos con Matlab. Curso 2016-17

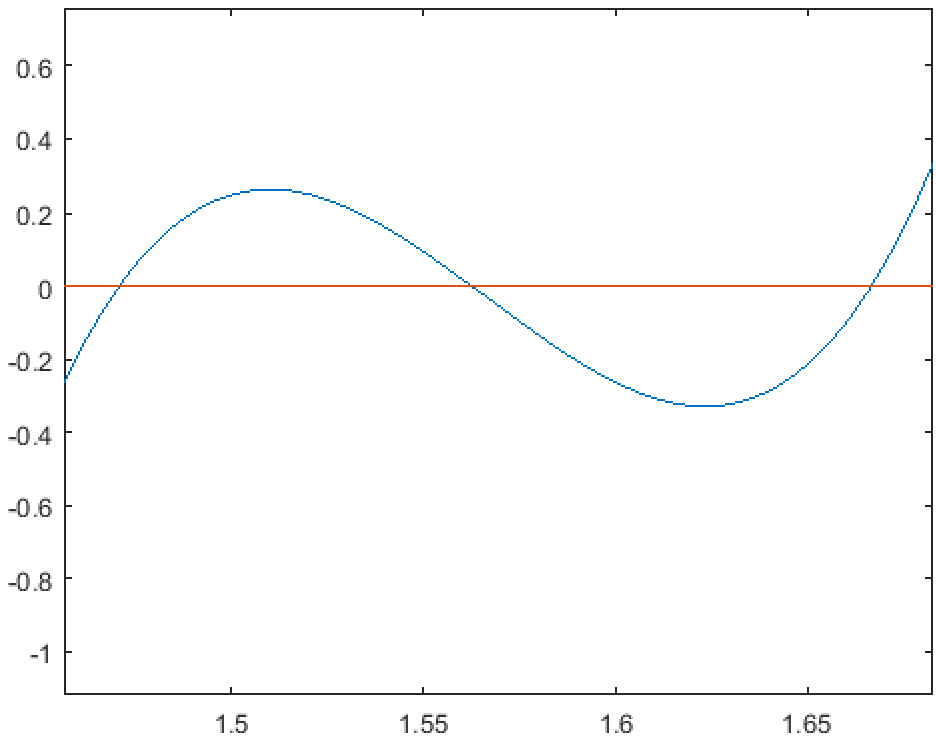
Eneko Olivares Gorriti, 04/05/2017

Tareas de las unidades 4, 5 y 6

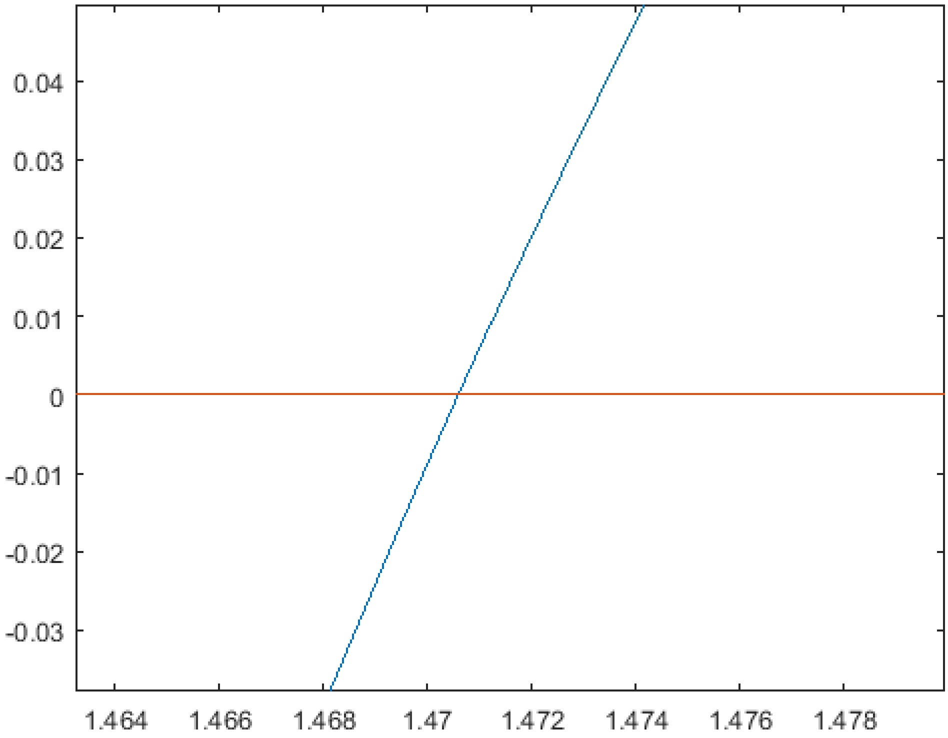
Ejercicio 1

Tras el estudio de la función podemos acotar que la función tiene las tres raíces más o menos en el intervalo que se muestra en la imagen acontinuación (hemos representado inicialmente en el intervalo `[1, 2]` y hemos hecho zoom hasta conseguir ver claramente el corte en el eje 0).

```
f = @(x) 816.*x.^3 - 3835.*x.^2 + 6000.*x - 3125;
fplot(f, [1, 2]);
hold on;
plot([1:0.1:2],0.*[1:0.1:2]);
```



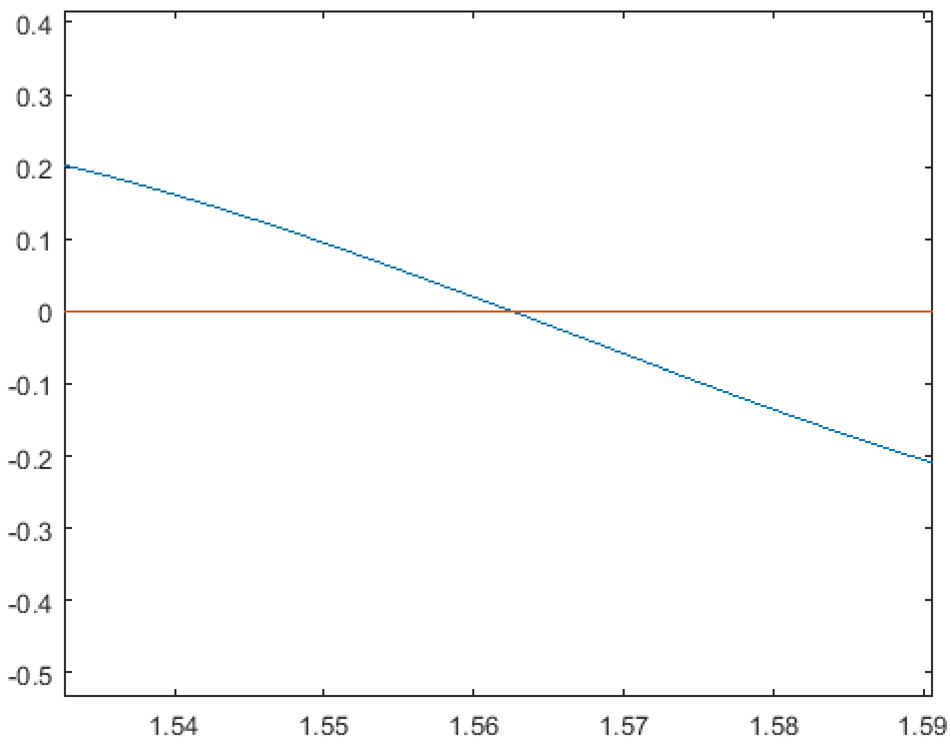
A continuación utilizaremos la función `fzero` en cada intervalo donde observemos que existe una raiz. El primer intervalo que contiene una raiz podemos acotarla, por ejemplo, entre `[1.464 1.472]` como podemos observar en la siguiente imagen:



Para obtener la raiz ejecutaremos el siguiente comando:

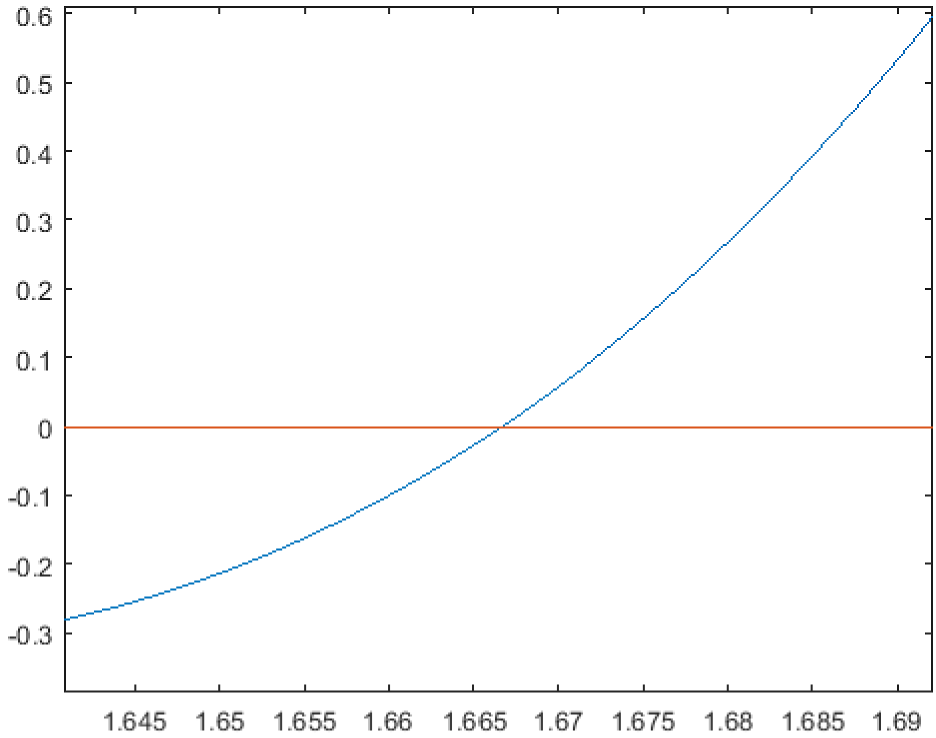
```
>> fzero(f, [1.464 1.472])

ans =

   1.470588235294055
```

Lo mismo para la segunda raiz en el intervalo `[1.54 1.58]`:



```
>> fzero(f, [1.54 1.58])

ans =

   1.562500000000050
```

Y para la tercer raiz en el intervalo `[1.65 1.68]`:



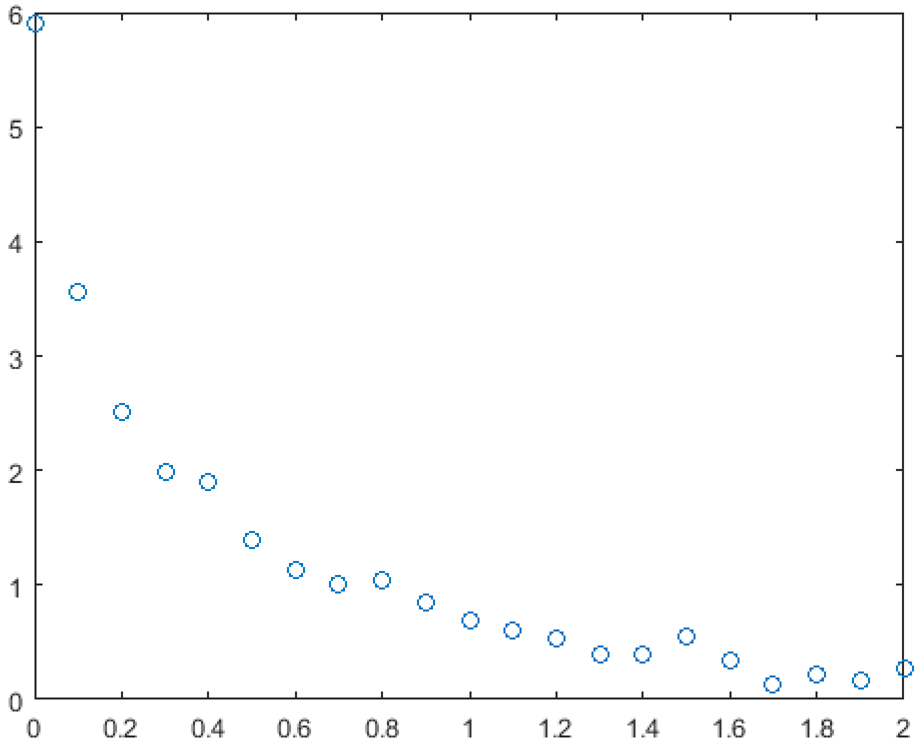

```
>> fzero(f, [1.65 1.68])

ans =

   1.666666666666668
```

Ejercicio 2

En el ejercicio se pide que dados unos datos muestreados se realicen diferentes ajustes minimo-cuadráticos y de interpolación.

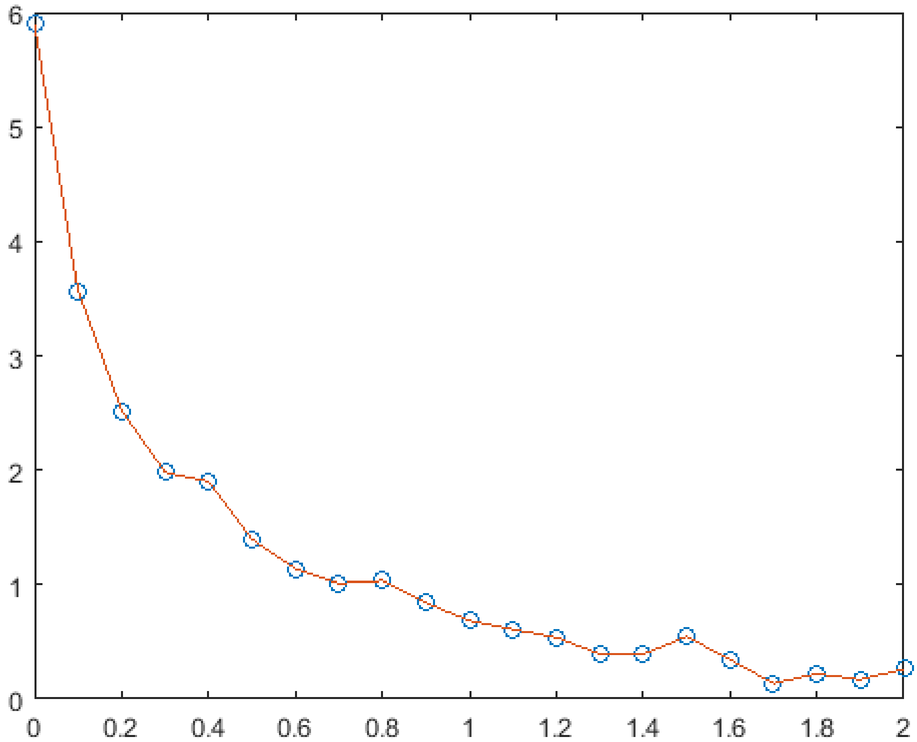
La muestra de datos se puede observar en la siguiente imagen:

```
t = (0:.1:2)';
y = [5.8955 3.5639 2.5173 1.9790 1.8990 1.3938 1.1359 ...
1.0096 1.0343 0.8435 0.6856 0.6100 0.5392 0.3946 ...
0.3903 0.5474 0.3459 0.1370 0.2211 0.1704 0.2636]';
plot(t, y, 'o');
```



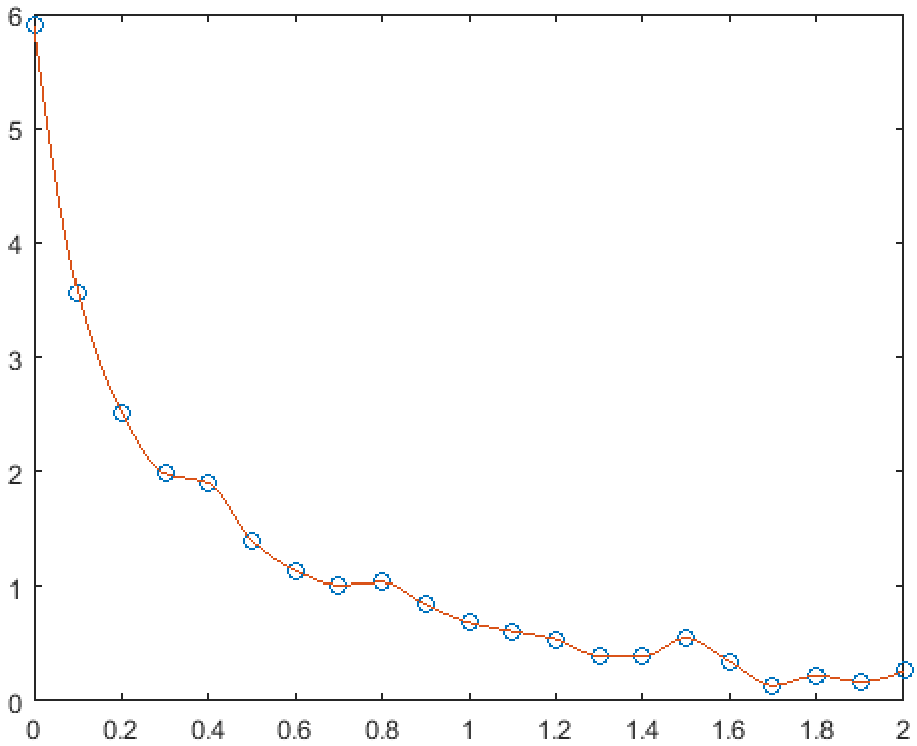
Interpolación lineal

```
xq = (0:0.01:2)';
yq = interp1(t, y, xq, 'linear');
plot(t, y, 'o', xq, yq);
```



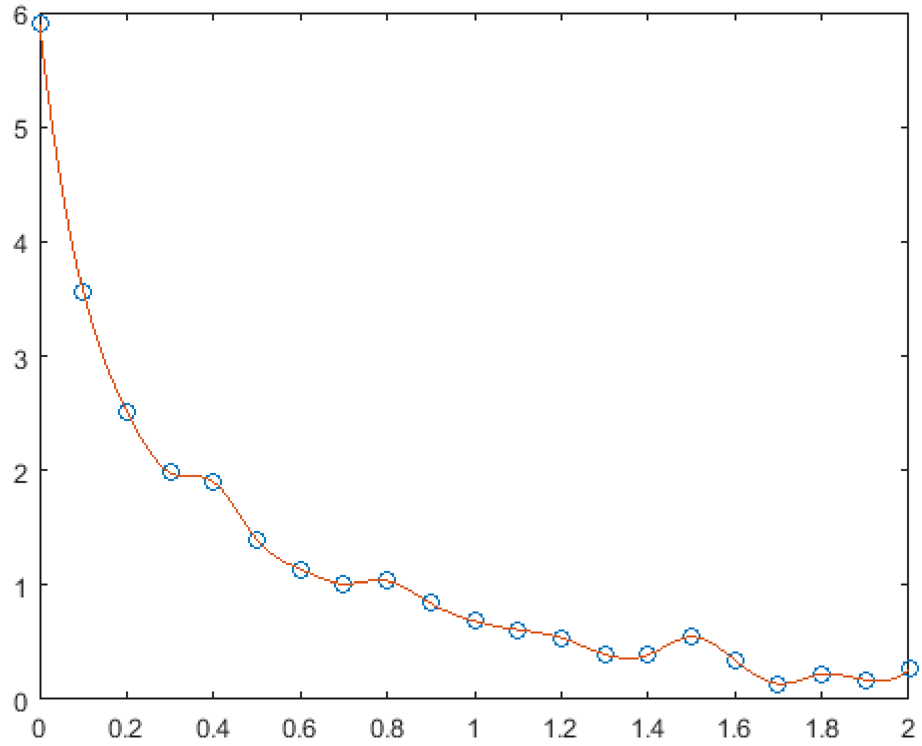
Interpolación a trozos con polinomios cúbicos de Hermite

```
xq = (0:0.01:2)';
yq = interp1(t, y, xq, 'pchip');
plot(t, y, 'o', xq, yq);
```



Interpolación a trozos con polinomios splines cúbicos

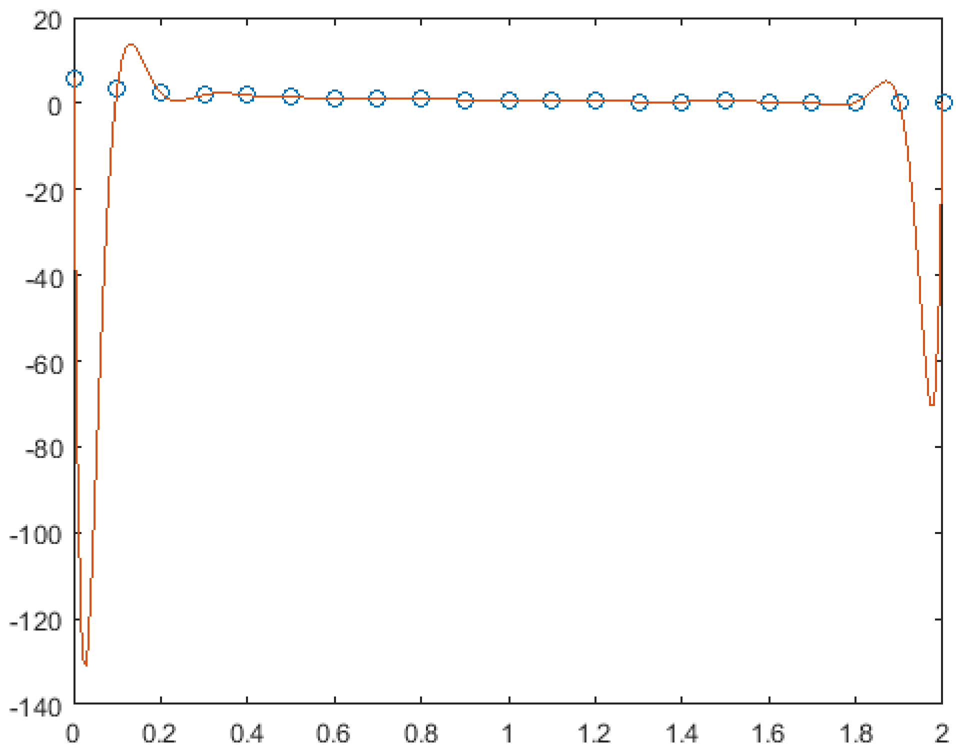
```
xq = (0:0.01:2)';
yq = interp1(t, y, xq, 'spline');
plot(t, y, 'o', xq, yq);
```



Interpolación polinómica

Como hay 21 valores de y, vamos a necesitar un polinomio de grado 20. Vamos a realizar primero un ajuste del eje `X` para que al evaluar sobre un polinomio de grado tan alto no ocurran problemas. El valor de `t` ajustado es la variable `tq = t - 1` y el valor de `xq` ajustado es `sq = xq - 1`, en la variable `P` guardaremos los coeficientes del polinomio.

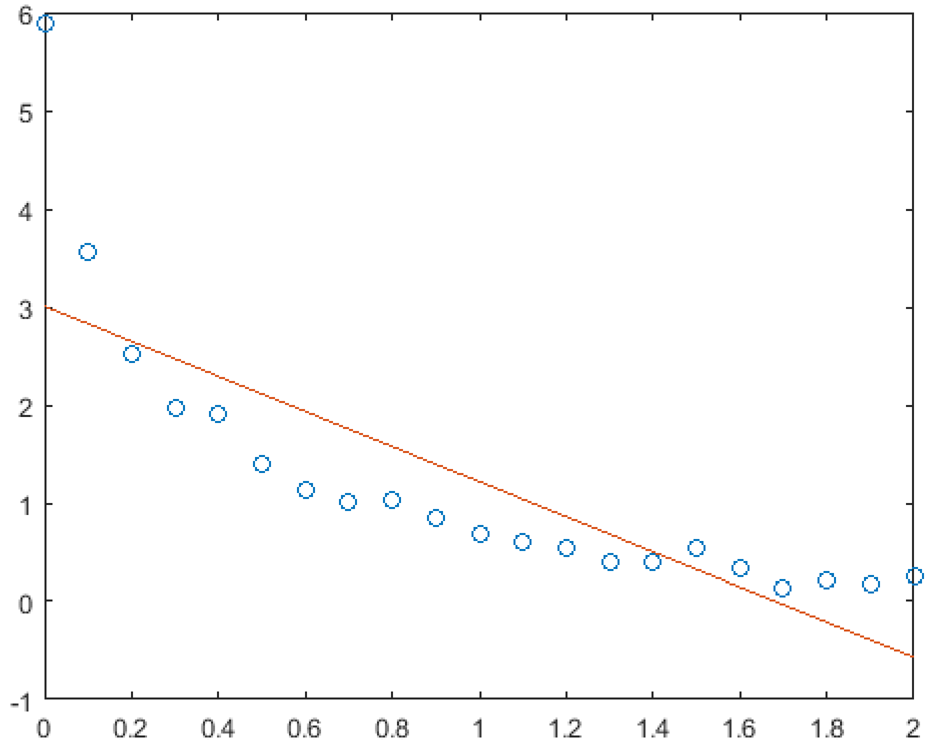
```
xq = (0:0.01:2)';
tq = t - 1;
sq = xq - 1;
P = polyfit(tq, y, 20);
yq = polyval(P, sq);
plot(t, y, 'o', xq, yq);
```



Ajuste lineal de los datos

Realizaremos un ajuste mínimo-cuadrático para encontrar los coeficientes (variable `b`) del polinomio de grado 1 que minimiza el error cuadrático entre los valores reales y aproximados.

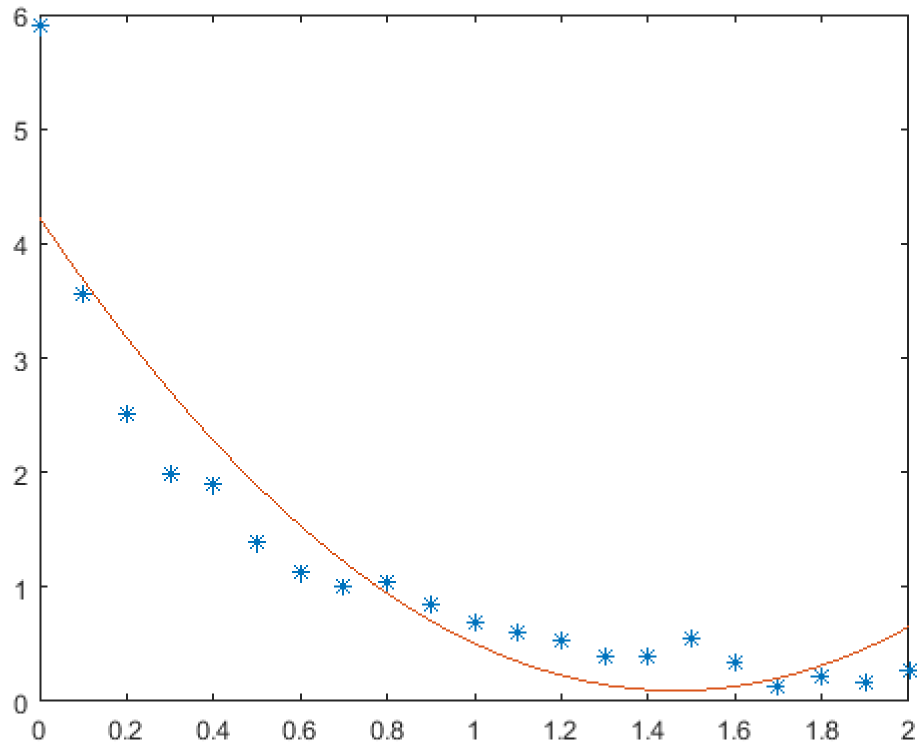
```
xq = (0:0.01:2)';
X = [t.^0 t.^1];
b = X\y;
yq = (b'*[xq.^0 xq.^1]')';
plot(t, y, 'o', xq, yq);
```



Ajuste cuadrático de los datos

Lo mismo que en el ajuste lineal pero esta vez con un polinomio de grado 2.

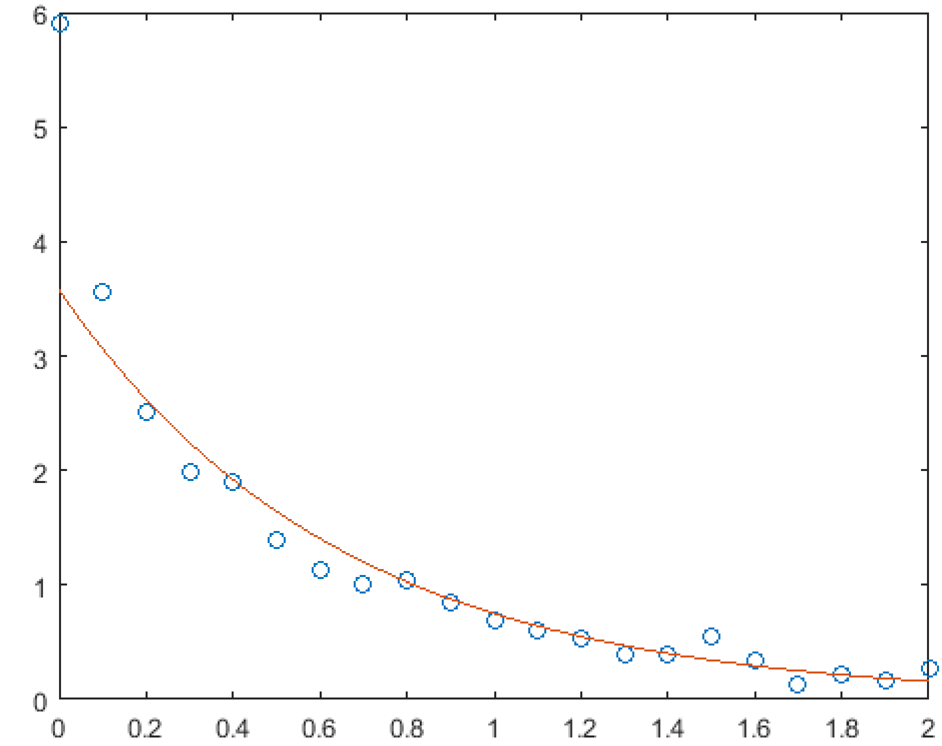
```
xq = (0:0.01:2)';
X = [t.^0 t.^1 t.^2];
b = X\y;
yq = (b'*[xq.^0 xq.^1 xq.^2]')';
plot(t, y, 'o', xq, yq);
```



Ajuste log-lineal de los datos

Transformaremos los datos para linealizar la función exponencial tomando logaritmos.

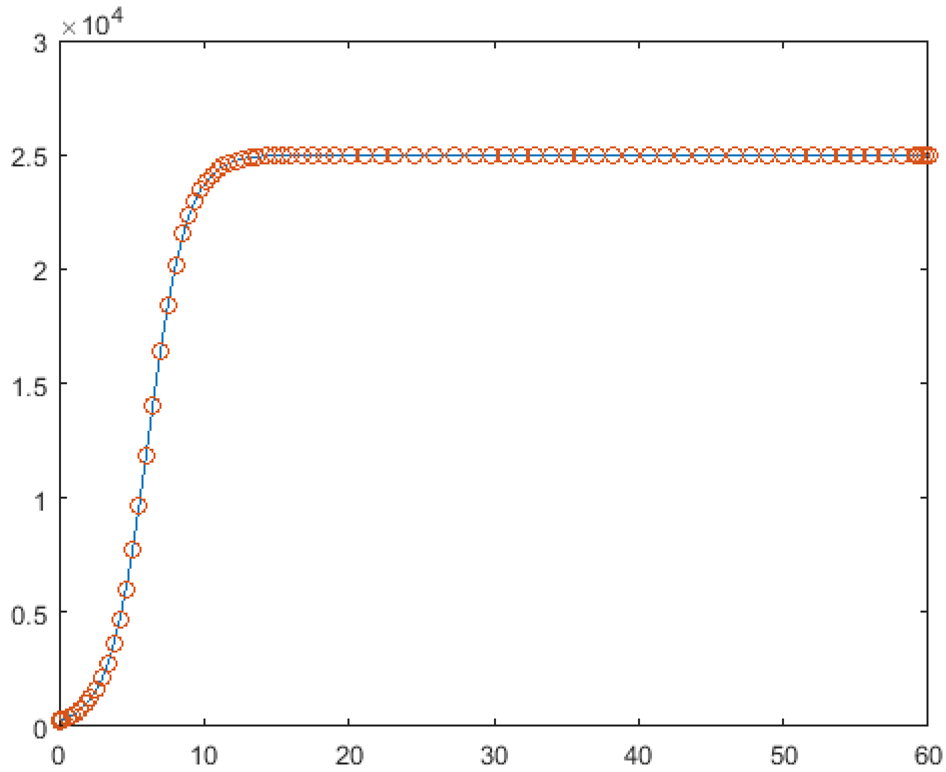
```
xq = (0:0.01:2)';
X=[t.^0 t.^1];
b=X\log(y);
yq=exp(b'*[xq.^0 xq.^1]')';
plot(t, y, 'o', xq, yq);
```



Ejercicio 3

Para calcular la evolución de la epidemia utilizaremos la función `ode45` para resolver la ecuación diferencial en el intervalo indicado.

```
f = inline('0.00003*y*(25000 - y)', 't', 'y');
[T, Y] = ode45(f, [0 60], 250);
plot(T,Y,T,Y,'o');
```



Calcular el número medio de personas contagiadas realizando la media aritmética:

```
media = sum(Y)/length(Y);

media =

   1.9244e+04
```

Ejercicio 4

Realizaremos el proceso completo de la sección de compresión de señales, para ello primero realizaremos los dos scripts necesarios: `cumenergy.m` y `rms.m`. A continuación se muestran el contenido de los script, y los comandos realizados para la obtención de las gráficas que muestran el proceso y los resultados obtenidos:

cumenergy.m

```
function [ y ] = cumenergy( x )

y = cumsum(x.^2)/(norm(x)^2);

end
```

rms.m

```
function [e] = rms(x,y)

  e = sqrt(norm(x-y)^2/length(x));

end
```

Proceso de compresión de la señal

```
t = linspace(0,4,2^12);
s = (30.*t.^2).*((2 - t).^5).*((4 - t).^2).*cos(24.*pi.*t) ...
 + 20.*(t.^2).*((2 - t).^2).*((4 - t).^5).*cos(12.*pi.*t);
[C,L] = wavedec(s,10,'coif5');
C_dec = abs(sort(-abs(C)));
ind_sobran = find(cumenergy(C_dec)>=0.9999);
umbral = C_dec(ind_sobran(1));
C_sig = wthresh(C,'h',umbral);
s_rec = waverec(C_sig,L,'coif5');
figure;
subplot(2,1,1);
plot(s);
axis([1 2^12 min(s) max(s)]);
title('original');
subplot(2,1,2);
plot(s_rec);
axis([1 2^12 min(s_rec) max(s_rec)]);
title('reconstruccion');
map = C_sig~=0;
val_sig = sum(map);
[comp,err] = sprintf('%d:%d',2^12,val_sig);
[comp_aprox,err] = sprintf('%d:%d',round(2^12/val_sig),1);
error = rms(s,s_rec);
[l_long_orig,err] = sprintf('Longitud original: %d \n',2^12);
[l_val_sig,err] = sprintf('Valores significativos: %d \n',val_sig);
```

```
[l_comp,err] = sprintf('Factor compresion de %s \n',comp);
[l_comp_aprox,err] = sprintf('\t (aproximadamente de %s) \n',comp_aprox);
[l_rms,err] = sprintf('Error RMS: %d \n',error);
sprintf('%s%s%s%s%s',l_long_orig,l_val_sig,l_comp,l_comp_aprox,l_rms)
```

Resultados

```
Longitud original: 4096
Valores significativos: 153
Factor compresion de 4096:153
     (aproximadamente de 27:1)
Error RMS: 1.516961e+01
```