

Computación

Aula Virtual: <https://famaf.aulavirtual.unc.edu.ar/course/view.php?id=747>

Resguardo tutoriales: <https://www.famaf.unc.edu.ar/~moreschi/docencia/Computacion/>

Tutorial Problemas 5 de la Guía N° 6

Problema 5: Determinación de la constante de un resorte de datos observacionales: Un bloque de masa $m = 1\text{kg}$, está sujeto a un resorte y puede deslizarse sobre una superficie horizontal libre de roce. El otro extremo del resorte está sujeto a una pared. Se realizan observaciones de la posición del centro de masa del bloque como función del tiempo. De estas mediciones se debe calcular la constante k del resorte.

En este problema nos enfrentamos a que para funciones que no son polinomios, el ajuste de funciones es más complicado. Intentaremos varias técnicas hasta conseguir un resultado aceptable.

Existe más de una forma de hacer el ajuste de curvas con funciones de PYTHON. En lo que sigue veremos el empleo de dos de ellas con variantes.

- a) Se simulará la situación experimental generando aleatoriamente datos alrededor de un movimiento ideal del bloque. Usaremos dos maneras alternativas de describir el movimiento teórico. Para ello use:

```
1 def f(t,a,om,fi):
2     return a*np.cos(om*t+fi)
3
4
5 """
6 Recordar que:
7 A cos(om*t+fi) = A cos(om*t) cos(fi) - A sin(om*t) sin(fi)
8 = cc cos(om*t) + cs sin(om*t)
9 tan(fi) = - cs/cc
10 A = sqrt( cc**2 + cs**2)
11 """
12 def f2(t,cc,cs,om):
13     return cc*np.cos(om*t) + cs*np.sin(om*t)
```

- b) El intervalo de tiempo, las constantes que determinan el movimiento y la generación de ruido observacional, se calculan de:

```
1 tmin = 0.
2 tmax = 14.
3
4 # elección de la semilla -----
5 np.random.seed(12)
6
7 # elección de Nrep -----
8 Nt = 1001
9 t = np.linspace(tmin, tmax, Nt)
10 AO = 5. # generamos datos con este valor ideal de A_0, om0 y fi0
11 om0 = 3.
12
13 cc = 4.
```

```

14 cs = 3.
15 fi0 = np.arctan(-cs/cc)
16
17 ancho = 2.
18
19 ya = f(t,A0,om0,fi0)
20 y = ya + np.random.normal(size=t.size,loc=0.,scale=ancho)
21
22 np.random.seed(12)
23 yb = f2(t,cc,cs,om0)
24 y2 = yb + np.random.normal(size=t.size,loc=0.,scale=ancho)

```

c) Genere un gráfico que muestre los datos observacionales.

d) Realizaremos varios ajustes con la función `curve_fit` de la librería `scipy.optimize`. En este caso usaremos:

```

1 popt1, pcov1 = curve_fit(f, t, y)
2 print('popt1 =',popt1)
3 print('pcov1 =',pcov1)
4
5 popt2, pcov2 = curve_fit(f2, t, yb)
6 print('popt2 =',popt2)
7 print('pcov2 =',pcov2)

```

e) Genere un gráfico mostrando estos ajustes y además fijando el rango de los posibles valores como se muestra en estos comandos:

```

1 plt.plot(t, f(t, *popt1), 'm-', linewidth=5.0,
2          label='fit: $A$=%6.4f, om=%6.4f, fi=%6.4f, (ajuste f)',
3          % tuple(popt1))
4
5 plt.plot(t, f2(t, *popt2), 'c--', linewidth=5.0,
6          label='fit: $Ac$=%6.4f, $As$=%6.4f, om=%6.4f, (ajuste f2)',
7          % tuple(popt2))
8
9 # Se puede forzar la búsqueda entre límites con "bounds":
10 popt, pcov = curve_fit(f, t, y, bounds=([3.5, 1, -2.3],
11                                       [6., 5., 2.14159]))
12 print('popt=',popt)
13 plt.plot(t, f(t, *popt), 'r-', linewidth=5.0,
14          label='fit: $A$=%6.4f, om=%6.4f, fi=%6.4f, (segundo ajuste f)',
15          % tuple(popt))
16
17 poptb, pcovb = curve_fit(f2, t, y2, bounds=([2.8, 2.7, 2.1],
18                                       [5.6, 5.6, 4.14159]))
19 print('poptb=',poptb)
20 plt.plot(t, f2(t, *poptb), 'g--', linewidth=5.0,
21          label='fit: $Ac$=%6.4f, $As$=%6.4f, om=%6.4f, (segundo ajuste f2)',
22          % tuple(poptb))

```

f) Realizaremos ahora varios ajustes con la función `least_squares` de la librería `scipy.optimize`. En este caso usaremos:

```

1 t_train = t
2 y_train = y
3
4 # notar que en lo que sigue x denota el arreglo de
5 # los coeficientes que se desean ajustar
6 def fun(x, t, y):
7     return x[0]*np.cos(x[1]*t+x[2]) - y
8
9 # propuesta de valores iniciales
10 x0 = np.array([3.5, 2.1, 0.])
11 print('    x0=',x0)
12
13 print()
14 print('        ejecuto: res_lsq = least_squares(fun, \
15 x0, args=(t_train, y_train), method="lm"')
16 res_lsq = least_squares(fun, x0, args=(t_train, y_train), method='lm')
17
18 print(' valores calculados:')
19 print('        res_lsq.x=',res_lsq.x)
20
21 print()
22 print(' otras dos formas usando valores para "loss":')
23
24 #res_soft_l1 = least_squares(fun, x0, loss='soft_l1', f_scale=0.1,
25 res_soft_l1 = least_squares(fun, x0, loss='huber', f_scale=0.1,
26 args=(t_train, y_train))
27 print('        res_soft_l1.x=',res_soft_l1.x)
28 print()
29 #res_log = least_squares(fun, x0, loss='cauchy', f_scale=0.1,
30 res_log = least_squares(fun, x0, loss='arctan', f_scale=0.1,
31 args=(t_train, y_train))
32 print('        res_log.x=',res_log.x)
33 print()

```

g) Genere un gráfico mostrando estos ajustes y además fijando el rango de los posibles valores como se muestra en estos comandos:

```

1 y_lsq = f(t, *res_lsq.x)
2 y_soft_l1 = f(t, *res_soft_l1.x)
3 y_log = f(t, *res_log.x)
4
5 plt.figure( figsize=(10, 7.5) )
6 plt.title('Observaciones + ajuste (f, least_squares) para el oscilador, '
7         +str(Nt)+' puntos')
8 plt.plot(t_train, y_train, 'o', label='observaciones')
9 plt.plot(t, y_lsq, linewidth=5.0,
10         label='fit: $A$=%6.4f, om=%6.4f, fi=%6.4f, linear loss, \
11         con: method="lm" '
12         % tuple(res_lsq.x))
13 plt.plot(t, y_soft_l1, linewidth=5.0,
14         label='fit: $A$=%6.4f, om=%6.4f, fi=%6.4f, huber loss '
15         % tuple(res_soft_l1.x))
16 plt.plot(t, y_log, linewidth=5.0,
17         label='fit: $A$=%6.4f, om=%6.4f, fi=%6.4f, arctan loss '
18         % tuple(res_log.x))

```

```

19 plt.plot(t, ya, 'k', linewidth=1.5, label='exacta')
20 plt.xlabel("t")
21 plt.ylabel("y")
22 plt.grid()
23 plt.legend(loc="best")
24 plt.savefig('graficos/p5-least_squares-observaciones-oscilador-',
25             +str(Nt)+'.png', dpi=100)
26 plt.show()

```

- h) Elija los valores calculados con alguno de los métodos y calcule la constante k del resorte.
- i) Corra el programa nuevamente con $Nt = 1000$ y realice los ajustes nuevamente.

Tutorial:

- Guarde en el archivo p5.py las siguientes instrucciones:

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 from scipy.optimize import curve_fit, least_squares, leastsq
4
5
6 def f(t,a,om,fi):
7     return a*np.cos(om*t+fi)
8
9 """
10 Recordar que:
11 A cos(om*t+fi) = A cos(om*t) cos(fi) - A sin(om*t) sin(fi)
12                = cc cos(om*t) + cs sin(om*t)
13 tan(fi) = - cs/cc
14 A = sqrt( cc**2 + cs**2)
15 """
16
17 def f2(t,cc,cs,om):
18     return cc*np.cos(om*t) + cs*np.sin(om*t)
19
20
21 # se asume que la unidad de tiempo es el segundo y la de posición el metro
22 tmin = 0.
23 tmax = 14.
24
25 # elección de la semilla -----
26 np.random.seed(12)
27
28 # elección de Nrep -----
29 Nt = 1001
30 t = np.linspace(tmin, tmax, Nt)
31 AO = 5. # generamos datos con este valor ideal de A_0, om0 y fi0
32 om0 = 3.
33
34 cc = 4.
35 cs = 3.
36 fi0 = np.arctan(-cs/cc)
37

```

```

38 ancho = 2.
39
40 ya = f(t,A0,om0,fi0)
41 y = ya + np.random.normal(size=t.size,loc=0.,scale=ancho)
42
43 np.random.seed(12)
44 yb = f2(t,cc,cs,om0)
45 y2 = yb + np.random.normal(size=t.size,loc=0.,scale=ancho)
46
47 print('t =',t)
48 print('y =',y)
49 print('len(t) =',len(t))
50 print('len(y) =',len(y))
51 print('min(y) =',min(y))
52 print('max(y) =',max(y))
53 print('    A0 =',A0)
54 print('    om0 =',om0)
55 print('    fi0 =',fi0)
56 print(' ancho =',ancho)
57 print('    cc =',cc)
58 print('    cs =',cs)
59
60
61 plt.figure(figsize=(10, 7.5) )
62 plt.title('Observaciones para el oscilador, '+str(Nt)+' puntos')
63 plt.xlabel('t')
64 plt.ylabel('y')
65 plt.grid()
66 plt.plot(t , y,'o-', label='observaciones')
67 plt.legend(loc="best")
68 plt.savefig('graficos/p5-observaciones-oscilador-'+str(Nt)+'.png', dpi=100)
69 plt.show()
70
71 print('----- Usando: curve_fit -----')
72
73 popt1, pcov1 = curve_fit(f, t, y)
74 print('popt1 =',popt1)
75 print('pcov1 =',pcov1)
76
77 popt2, pcov2 = curve_fit(f2, t, yb)
78 print('popt2 =',popt2)
79 print('pcov2 =',pcov2)
80
81
82 plt.figure(figsize=(10, 7.5) )
83 plt.title('Observaciones + ajuste (f y f2, curve_fit) para el oscilador, '
84         +str(Nt)+' puntos')
85 plt.xlabel('t')
86 plt.ylabel('y')
87 plt.grid()
88 plt.plot(t , y,'o', label='observaciones')
89 plt.plot(t, f(t, *popt1), 'm-', linewidth=5.0,
90         label='fit: $A$=%6.4f, om=%6.4f, fi=%6.4f, (ajuste f)'
91         % tuple(popt1))
92
93 plt.plot(t, f2(t, *popt2), 'c--', linewidth=5.0,

```

```

94     label='fit: $Ac$=%6.4f, $As$=%6.4f, om=%6.4f, (ajuste f2)'
95     % tuple(popt2))
96
97 # Se puede forzar la búsqueda entre límites con "bounds":
98 popt, pcov = curve_fit(f, t, y, bounds=([3.5, 1, -2.3],
99                                         [6., 5., 2.14159]))
100 print('popt=', popt)
101 plt.plot(t, f(t, *popt), 'r-', linewidth=5.0,
102          label='fit: $A$=%6.4f, om=%6.4f, fi=%6.4f, (segundo ajuste f)'
103          % tuple(popt))
104
105 poptb, pcovb = curve_fit(f2, t, y2, bounds=([2.8, 2.7, 2.1],
106                                             [5.6, 5.6, 4.14159]))
107 print('poptb=', poptb)
108 plt.plot(t, f2(t, *poptb), 'g--', linewidth=5.0,
109          label='fit: $Ac$=%6.4f, $As$=%6.4f, om=%6.4f, (segundo ajuste f2)'
110          % tuple(poptb))
111
112 plt.plot(t, ya, 'k', linewidth=1.5, label='exacta')
113 plt.legend(loc="best")
114 plt.savefig('graficos/p5-observaciones+ajuste-oscilador-'+str(Nt)+
115            '.png', dpi=100)
116 plt.show()
117
118
119 print('----- Usando: least_squares -----')
120
121 t_train = t
122 y_train = y
123
124 # notar que en lo que sigue x denota el arreglo de
125 # los coeficientes que se desean ajustar
126 def fun(x, t, y):
127     return x[0]*np.cos(x[1]*t+x[2]) - y
128
129 # propuesta de valores iniciales
130 x0 = np.array([3.5, 2.1, 0.])
131 print('    x0=', x0)
132
133 print()
134 print('    ejecuto: res_lsq = least_squares(fun, \
135 x0, args=(t_train, y_train), method="lm")')
136 res_lsq = least_squares(fun, x0, args=(t_train, y_train), method='lm')
137
138 print(' valores calculados:')
139 print('    res_lsq.x=', res_lsq.x)
140 #print()
141 print('    res_lsq.cost=', res_lsq.cost)
142 #print()
143 print(' res_lsq.optimality=', res_lsq.optimality)
144
145 print()
146 print(' otras dos formas usando valores para "loss":')
147
148 #res_soft_l1 = least_squares(fun, x0, loss='soft_l1', f_scale=0.1,
149 res_soft_l1 = least_squares(fun, x0, loss='huber', f_scale=0.1,

```

```

150         args=(t_train, y_train))
151 print('      res_soft_l1.x=',res_soft_l1.x)
152 print()
153 #res_log = least_squares(fun, x0, loss='cauchy', f_scale=0.1,
154 res_log = least_squares(fun, x0, loss='arctan', f_scale=0.1,
155         args=(t_train, y_train))
156 print('      res_log.x=',res_log.x)
157 print()
158
159 """
160 Graficamos las distintas curvas obtenidas.
161 """
162
163 y_lsq = f(t, *res_lsq.x)
164 y_soft_l1 = f(t, *res_soft_l1.x)
165 y_log = f(t, *res_log.x)
166
167 plt.figure( figsize=(10, 7.5) )
168 plt.title('Observaciones + ajuste (f, least_squares) para el oscilador, '
169 +str(Nt)+' puntos')
170 plt.plot(t_train, y_train, 'o', label='observaciones')
171 plt.plot(t, y_lsq, linewidth=5.0,
172         label='fit: $A$=%6.4f, om=%6.4f, fi=%6.4f, linear loss, \
173 con: method="lm"')
174         % tuple(res_lsq.x))
175 plt.plot(t, y_soft_l1, linewidth=5.0,
176         label='fit: $A$=%6.4f, om=%6.4f, fi=%6.4f, huber loss'
177         % tuple(res_soft_l1.x))
178 plt.plot(t, y_log, linewidth=5.0,
179         label='fit: $A$=%6.4f, om=%6.4f, fi=%6.4f, arctan loss'
180         % tuple(res_log.x))
181 plt.plot(t, ya, 'k', linewidth=1.5, label='exacta')
182 plt.xlabel("t")
183 plt.ylabel("y")
184 plt.grid()
185 plt.legend(loc="best")
186 plt.savefig('graficos/p5-least_squares-observaciones-oscilador-'
187 +str(Nt)+'.png', dpi=100)
188 plt.show()
189
190 #quit()

```

- Desde la terminal ejecute:


```
python3 p5.py
```

 e interprete el resultado.
 Alternativamente ejecute:


```
python3
```

 y vaya agregando uno a uno los bloques del programa.
- Estudie cada paso del programa y agregue comentarios explicativos.
- Corra el programa nuevamente con $Nt = 1000$ y realice los ajustes nuevamente.

Se deberían generar los siguientes gráficos:

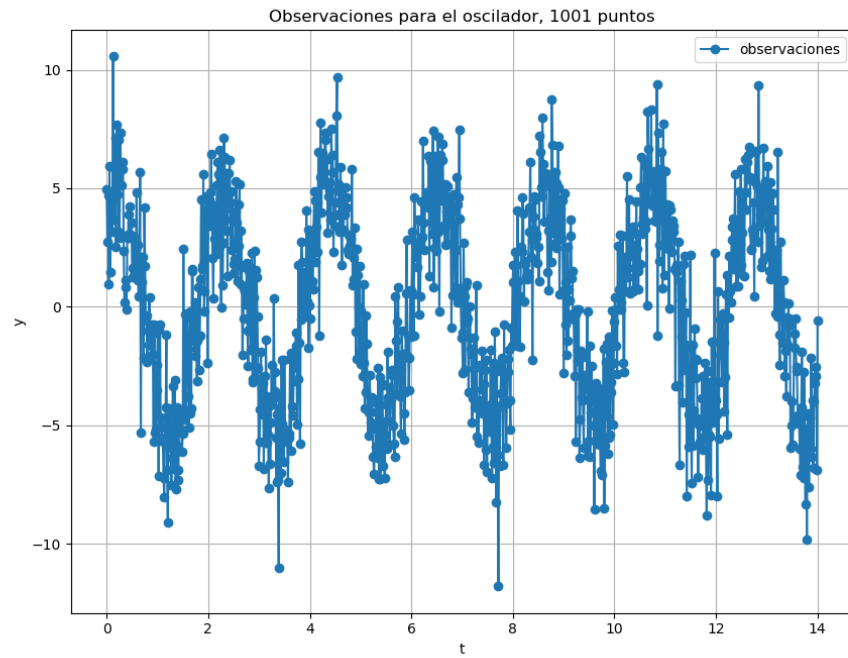


Figura 1: Observaciones para determinar la constante k del oscilador.

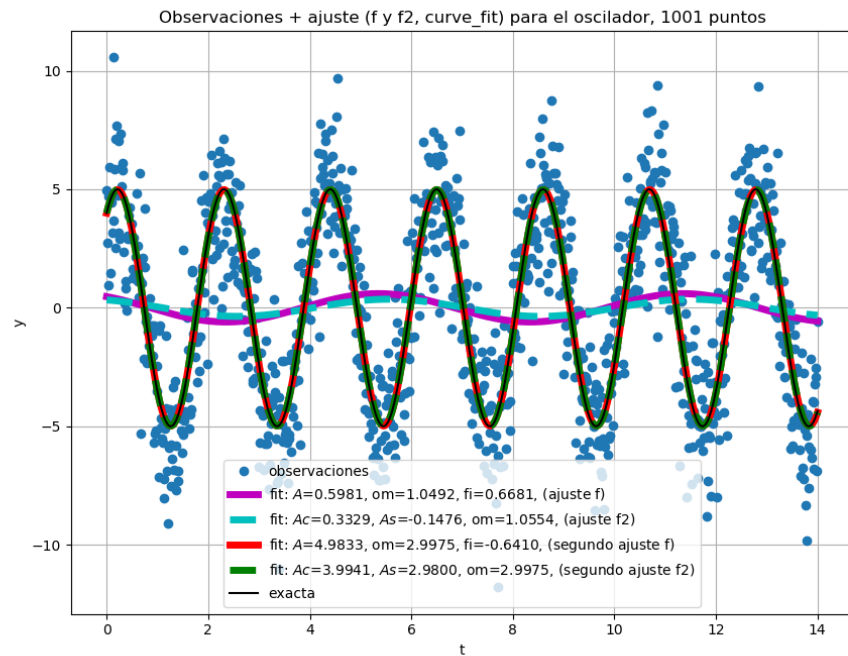


Figura 2: Observaciones y ajustes, usando 'curve_fit', para determinar la constante k .

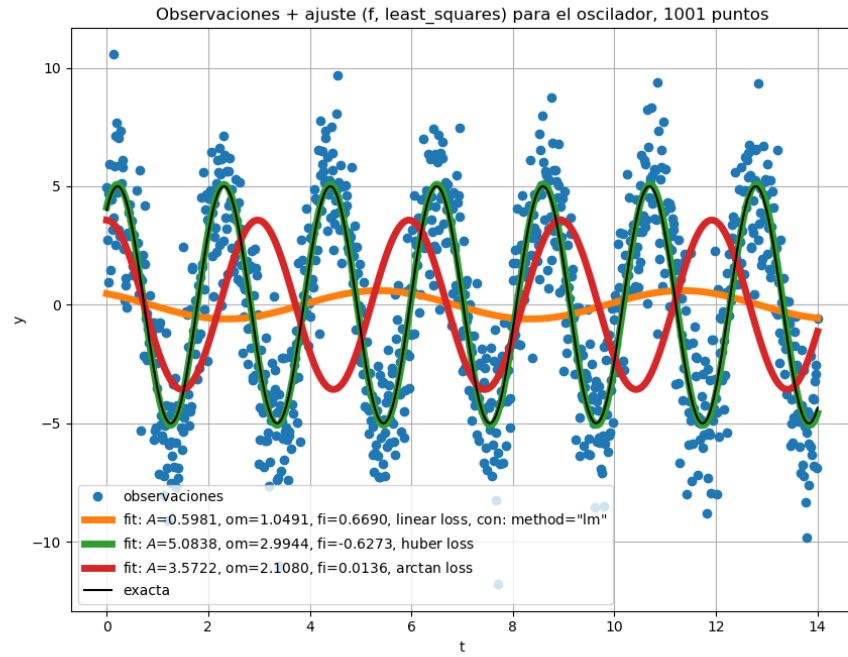


Figura 3: Observaciones y ajustes, usando 'least_squares' para determinar la constante k .