Apéndice A

Programas en Python

Programa A.1: Representación y datos. Variación de la temperatura de entrada de agua.

```
'''Programa para cargar y leer datos, representarlos
y obtener las constantes para identificar un
modelo  de la temperatura de salida del agua
respecto de la de entrada'''
import numpy as np
import matplotlib.pyplot as plt
#Inicializa variables
x_tp = 0
x_td = 0
i = 0
#Abre el archivo con los datos importados de Fluent y lee los datos
with open("step_t_e.txt") as f:
    datos = f.read()
#Divide las columas y las asigna a un vector
datos = datos.split('\n')
y_esc = [float(row.split(' ')[1])  for row in datos]
#Guarda en una variable la mitad de la longitud del vector de datos
len_data = len(data) / 2
#Guarda en dos vectores el tiempo y la salida escalados
y normalizados
x = [(int(row.split(' ')[0]) - 4) for row in datos]
y = [(float(row.split(' ')[1]) - y_esc[0]) for row in datos]
#Variables de salida max y min
y_max =  y[len_data]
y_min = y[0]
#Calcula de la salida que determina la constante de tiempo
y_tc = 0.632 * (y_max - y_min)
#Calcula el instante en el cual se alcanza la anterior salida
while y[i] < y_tc:
    i = i + 1
```

```
31      x_tc = x[i]
32  i = 1
33  #Calcula el retraso
34  while y[i]<=0.0001:
35      i = i + 1
36      x_td = x[i]
37  #Normaliza la constante de tiempo
38  x_tc_real = x_tc - x_td
39  #Representaciones
40  fig = plt.figure()
41  graf = fig.add_subplot(1,1,1)
42  graf.set_title('Sg 0: sube 1 K T.ent.
43  Sg %d: baja 1 K T.ent.' %len_data)
44  graf.set_xlabel('Segundos')
45  graf.set_ylabel('Respuesta grados K a la salida')
46  graf.axvline(x=x_tc, color='g',
47  label = 'Constante de tiempo = %d s' %x_tc_real)
48  graf.axvline(x=x_td, color='b',
49  label = 'Tiempo de retardo = %d s' %x_td)
50  graf.axvline(x=len_data, color='y',
51  label = 'Cambio de T entrada = %d s' %len_data)
52  graf.axhline(y = y_max, color='k',
53   label = 'Ganancia = %f ' %y_max)
54  graf.plot(x,y, c='r', label='Respuesta T salida')
55  leg = graf.legend()
56  #Imprime resultados por pantalla
57  print 'Tiempo de retardo: %d' %x_td
58  print 'Constante de tiempo: %d' %x_tc_real
59  print 'Ganancia: %f' %y_max
60  #Ajuste de plot
61  plt.xlim((0,2*len_data))
```

Programa A.2: Representación entrada escalón

```
1  '''Respuesta de g(s) ante entrada step '''
2  #se cargan las funciones necesarias
3  from control import matlab
4  import matplotlib.pyplot as plt
5  import sympy
6  from sympy import *
7  sympy.init_printing()
8  #def s
9  s = Symbol('s')
10 #def y argumentos
11 def stepResponse(ganancia,contante_tiempo,retraso,n):
```

```
12      Tf = ganancia/((1 + (contante_tiempo/n) * s)**n)
13      #def numerador
14      num = Poly(Tf.as_numer_denom()[0],s).all_coeffs()
15      #def denominador
16      den = Poly(Tf.as_numer_denom()[1],s).all_coeffs()
17      # f transferencia
18      tf = matlab.tf(map(float,num),map(float,den))
19      #respuesta step
20      t, y =matlab.step(tf)
21      #plot de la respuesta deplazado por el retraso
22      plt.plot((y+retraso),t)
23      plt.show()              #muestra de la figura
```

Programa A.3: Representación y datos. Variación de la temperatura de humos.

```
1  '''Programa para cargar y leer datos, representarlos
2  y obtener las constantes para identificar un modelo
3   de la temperatura de salida del agua respecto de la humos'''
4  import numpy as np
5  import matplotlib.pyplot as plt
6  x_tc = 0
7  x_td = 0
8  i = 0
9  with open("step_t_h.txt") as f:
10     data = f.read()
11 data = data.split('\n')
12 y_esc = [float(row.split(' ')[1])  for row in data]
13 len_data = len(data) / 2
14 x = [(int(row.split(' ')[0]) - 4) for row in data]
15 y = [(float(row.split(' ')[1]) - y_esc[0]) for row in data]
16 y_max =  y[len_data]
17 y_min = y[0]
18 y_tc = 0.632 * (y_max - y_min)
19 print y_tc
20 while y[i] > y_tc:
21     i = i + 1
22     x_tc = x[i]
23 print x_tc
24 i = 1
25 while y[i] > -0.0001:
26     i = i + 1
27     x_td = x[i]
28 x_tc_real = x_tc - x_td
29 fig = plt.figure()
30 graf = fig.add_subplot(1,1,1)
```

```
31 graf.set_title('Sg 0: suben 300 K T.ent. humos
32 Sg %d: bajan 300 K T.ent. humos' %len_data)
33 graf.set_xlabel('Segundos')
34 graf.set_ylabel('Respuesta grados K a la salida')
35 graf.axvline(x=x_tc, color='g',
36  label = 'Constante de tiempo = %d s' %x_tc_real)
37 graf.axvline(x=x_td, color='b',
38 label = 'Tiempo de retardo = %d s' %x_td)
39 graf.axvline(x=len_data, color='y',
40 label = 'Cambio de T entrada = %d s' %len_data)
41 graf.axhline(y = y_max, color='k',
42 label = 'Ganancia = %f ' %y_max)
43 graf.plot(x,y, c='r', label='Respuesta T salida')
44 leg = graf.legend(loc = 7)
45 print 'Tiempo de retardo: %d' %x_td
46 print 'Constante de tiempo: %d' %x_tc_real
47 print 'Ganancia: %f' %y_max
48 plt.ylim((-1,0.2))
```

Programa A.4: Representación de polos y ceros de una función

```
1 '''Programa para calcular y prepresentar polos y ceros de
2 una funcion de transferencia'''
3 from scipy import signal
4 import matplotlib.pyplot as plt
5 #Define como sys la funcion de transferencia
6 sys = signal.lti([0,1], [3306.25,115,1])
7 #Genera una figua
8 fig,  (pzmap) = plt.subplots(1)
9 #Muestra por pantalla polos y cero
10 print  ('Ceros = ' ,sys.zeros)
11 print ('Polos = ' ,sys.poles)
12 #Ajusta los limites para una mejor visualizacion
13 plt.xlim(xmax = 0.1, xmin = -1)
14 plt.grid(True)#Representa cuadricula
15 #Dibuja polos y ceros con distintos simbolos
16 pzmap.plot(sys.zeros.real, sys.zeros.imag, 'o')
17 pzmap.plot(sys.poles.real, sys.poles.imag, 'x')
```

Programa A.5: Representación de una señal binaria aleatoria para ser usada como variación de la temperatura de entrada de agua

```
1 '''Programa para la representacion de una sucesion
2 binaria aletoria para su utilizacion como variacion
3 de la temperatura de agua'''
4 import matplotlib.pyplot as plt
```

```
import numpy as np
#definicion de funcion lineas
def lineas(ax, pos, *args, **kwargs):
    for p in pos:
        plt.axvline(p, *args, **kwargs)
#array con la sucesion obtenida mediante
funcion nativa de Python
bits = [1,0,1,0,1,0,0,0,1,0,1,0,1,1,1,0,1,1,1,0]
pos_linea = []
for i in range(20):
    pos_linea.append (400*i)
#400 segundos entre cada cambio de bit
data = np.repeat(bits, 1)
t = 400 * np.arange(len(data))
lineas('x', pos_lines, color='.5', linewidth=1)
plt.step(t, data, 'r', linewidth = 1, where='post')
plt.ylim([-0.1,1.2])
#Escritura del segundo correpondiente a cada bit en la figura
for pos_bit, bit in enumerate(bits):
    plt.text(180 + pos_bit*400, -0.08, str(bit))
```

Programa A.6: Representación de la variación de la temperatura de salida del agua en función de la temperatura de entrada según una señal binaria aleatoria

```
'''Programa para la representacion de la respuesta
de la temperatura de salida ante cambios de la de
entrada segun una sucesion binaria aleatoria'''
import numpy as np
import matplotlib.pyplot as plt
#inicializaicon de variables
x_tp = 0
x_td = 0
i = 0
#Abre archivo con los datos obtenidos en Fluent
with open("digital_te.txt") as f:
    data = f.read()
#Separa datos por columnas y guarda los datos de
salida en y_esc
data = data.split('\n')
y_esc = [float(row.split(' ')[1])  for row in data]
len_data = len(data) / 2
#Guarda los tiempos en x y la T salida en y normalizada
x = [((int(row.split(' ')[0])*10) - 4) for row in data]
y = [(float(row.split(' ')[1]) - y_esc[0]) for row in data]
y_max =  y[len_data]
```

```
22 y_min = y[0]
23 fig = plt.figure()
24 graf1 = fig.add_subplot(1,1,1)
25 graf1.set_title('Bit = 1, cambia 1 grado entrada.
26 Bit = 0, mantiene')
27 graf1.set_xlabel('Segundos')
28 graf1.set_ylabel('Respuesta grados K a la salida')
29 graf1.plot(x,y, c='b', label='Respuesta T salida')
30 leg = graf1.legend()
31 t_i = []
32 #Representacion de cada segundo en el eje x
33 for i in range(20):
34     t_i.append(y[40*i])
35 print t_i
36 plt.xlim((0,2*len_data*10))
```

Programa A.7: Representación de una señal binaria aleatoria como variación a la temperatura de entrada de humos

```
1  '''Programa para la representacion de una sucesion
2  binaria aletoria para su utilizacion como variacion
3  de la temperatura de humos a su entrada'''
4  import matplotlib.pyplot as plt
5  import numpy as np
6  def my_lines(ax, pos, *args, **kwargs):
7      for p in pos:
8          plt.axvline(p, *args, **kwargs)
9  bits = [1,0,0,0,1,1,1,0,0,0,0,1,1,1,1,0,0,1,0,1]
10 pos_lines = []
11 for i in range(20):
12     pos_lines.append (500*i)
13 data = np.repeat(bits, 1)
14 t = 500 * np.arange(len(data))
15 my_lines('x', pos_lines, color='.5', linewidth=1)
16 plt.step(t, (data-1), 'r', linewidth = 1.5, where='post')
17 for pos_bit, bit in enumerate(bits):
18     plt.text(200 + pos_bit*500, -0.08, str(bit))
```

Programa A.8: Representación de la variación de la temperatura de salida del agua en función de la temperatura de entrada de humos según una señal binaria aleatoria

```
1  '''Programa para la representacion de la respuesta
2  de la temperatura de salida ante cambios de la de
3  entrada de humos segun una sucesion binaria aleatoria'''
4  import numpy as np
```

```
 5 import matplotlib.pyplot as plt
 6 x_tp = 0
 7 x_td = 0
 8 i = 0
 9 with open("digital_th.txt") as f:
10     data = f.read()
11 data = data.split('\n')
12 y_esc = [float(row.split(' ')[1])  for row in data]
13 len_data = len(data) / 2
14 x = [((int(row.split(' ')[0])*10) - 4) for row in data]
15 y = [(float(row.split(' ')[1]) - y_esc[0]) for row in data]
16 y_max =  y[len_data]
17 y_min = y[0]
18 fig = plt.figure()
19 graf1 = fig.add_subplot(1,1,1)
20 graf1.set_title('Bit = 1, cambia 300 grados entrada humos.
21 Bit = 0, mantiene')
22 graf1.set_xlabel('Segundos')
23 graf1.set_ylabel('Respuesta grados K a la salida')
24 graf1.plot(x,y, c='b', label='Respuesta T salida')
25 leg = graf1.legend()
26 t_i = []
27 for i in range(20):
28     t_i.append(y[40*i])
29 print t_i
30 plt.ylim(-1.1,0.1)
31 plt.xlim((0,2*len_data*10))
```

Programa A.9: Modelo de indentificacion discreto con un tiempo de discretizacion de 10 sg de la T de agua a la salida respecto a la de entrada

```
 1 '''Calculo del modelo de indentificacion discreto con un tiempo
 2 de discretizacion de 10 sg de la T de agua
 3  a la salida respecto a los cambios de la de entrada.
 4 Representacion repecto a resultados experimentales'''
 5 import numpy as np
 6 import matplotlib.pyplot as plt
 7 time, temperature, input = np.loadtxt('digital10_te.txt',
 8 unpack = True)
 9 time = (time-3)*10
10 a1 = []
11 a2 = []
12 b1 = []
13 b2 = []
14 y = []
```

```
15 #Calcula de la matriz con los regresores
16 i = 9
17 while i < 299:
18     a1.append(temperature[i])
19     i = i +1
20 j = 8
21 while j < 298:
22     a2.append(temperature[j])
23     j = j +1
24 k = 1
25 while k < 291:
26     b1.append(input[k])
27     k = k+1
28 l = 0
29 while l < 290:
30     b2.append(input[l])
31     l = l +1
32 m = 10
33 while m <=299:
34     y.append(temperature[m])
35     m = m +1
36 y = np.matrix(y)
37 y = np.transpose(y)
38 #Calculo de la matriz phi
39 phi_t= [a1, a2, b1, b2]
40 phi = np.transpose(phi_t)
41 phi = np.matrix(phi)
42 phi_t = np.matrix(phi_t)
43 pr_phi =  phi_t * phi
44 inv = np.linalg.inv(pr_phi)
45 inv = np.matrix(inv)
46 #Calculo de la matriz tita
47 tita_pre = inv * phi_t
48 tita = tita_pre * y
49 #Modelo obtenido respecto a datos experimentales
50 u =  [0]*70+ [1]*160+ [0]*80+ [1]*80+[0]
51 *40+[1]*40 + [0]*40 + [0]*40 + [1]*40 + [0]*20
52 z = 10
53 resultd = []
54 time, temperature= np.loadtxt('digital_te.txt', unpack = True)
55 while (z < 611):
56     y_est  = tita[0][0,0] * temperature[z + 88] + tita[1][0,0]
57     * temperature[z + 87]  + tita[2][0,0] * u[z-8]
58      + tita[3][0,0]*u[z-9]
```

```
59      resultd.append(y_est)
60      z = z + 1
61  resultd = np.resize(resultd,(1,600))
62  resultd = resultd.flatten() - 340.3963623046875
63  t = np.arange(1020,7020,10)
64  plt.plot(t,resultd, linewidth = 2, label = 'y estimada cada 10 s')
65  leg = plt.legend(loc = 3)
```

Programa A.10: Modelo de indentificacion discreto con un tiempo de discretizacion de 20 sg de la T de agua a la salida respecto a la de entrada

```
1  '''Calculo del modelo de indentificacion discreto con un tiempo
2  de discretizacion de 20 sg de la T de agua a la
3  salida respecto a los cambios de la de entrada.
4  Representacion repecto a resultados experimentales'''
5  import numpy as np
6  import matplotlib.pyplot as plt
7  time, temperature, input = np.loadtxt('digital10_te.txt', unpack = True)
8  time = (time-3)*10
9  a1 = []
10 a2 = []
11 b1 = []
12 b2 = []
13 y = []
14 temperature = temperature[0::2]
15 input = input[0::2]
16 i = 5
17 while i < 149:
18      a1.append(temperature[i])
19      i = i +1
20 j = 4
21 while j < 148:
22      a2.append(temperature[j])
23      j = j +1
24 k = 2
25 while k < 146:
26      b1.append(input[k])
27      k = k+1
28 l = 1
29 while l < 145:
30      b2.append(input[l])
31      l = l +1
32 m = 6
33 while m <=149:
34      y.append(temperature[m])
```

```
35      m = m +1
36  y = np.matrix(y)
37  y = np.transpose(y)
38  phi_t= [a1, a2, b1, b2]
39  phi = np.transpose(phi_t)
40  phi = np.matrix(phi)
41  phi_t = np.matrix(phi_t)
42  pr_phi =  phi_t * phi
43  inv = np.linalg.inv(pr_phi)
44  inv = np.matrix(inv)
45  tita_pre = inv * phi_t
46  tita = tita_pre * y
47  u = [0]*70+ [1]*160+ [0]*80+ [1]*80+[0]*40+
48  [1]*40 + [0]*40 + [0]*40 + [1]*40 + [0]*20
49  u = u[0::2]
50  z = 5
51  resultd = []
52  time, temperature= np.loadtxt('digital_te.txt', unpack = True)
53  temperature = temperature[0::2]
54  while (z < 309):
55      y_est  = tita[0][0,0] * temperature[z + 44] + tita[1][0,0]
56      * temperature[z + 43]  + tita[2][0,0] * u[z-4] + tita[3][0,0]*u[z-5]
57      resultd.append(y_est)
58      z = z + 1
59  resultd = np.resize(resultd,(1,300))
60  resultd = resultd.flatten() - 340.3963623046875
61  t = np.arange(1040,7040,20)
62  plt.plot(t,resultd, linewidth = 2,
63  label = 'y estimada cada 20 s')
64  leg = plt.legend(loc = 3)
```

Programa A.11: Modelo de indentificacion discreto con un tiempo de discretizacion de 40 sg de la T de agua a la salida respecto a la de entrada

```
1  '''Calculo del modelo de indentificacion discreto con un tiempo
2  de discretizacion de 40 sg de la T de agua a la
3  salida respecto a los cambios de la de entrada.
4  Representacion repecto a resultados experimentales'''
5  import numpy as np
6  import matplotlib.pyplot as plt
7  time, temperature, input = np.loadtxt('digital10_te.txt',
8  unpack = True)
9  time = (time-3)*10
10 a1 = []
11 a2 = []
```

```
b1 = []
b2 = []
y = []
temperature = temperature[0::4]
input = input[0::4]
i = 3
while i < 74:
    a1.append(temperature[i])
    i = i +1
j = 2
while j < 73:
    a2.append(temperature[j])
    j = j +1
k = 1
while k < 72:
    b1.append(input[k])
    k = k+1
l = 0
while l < 71:
    b2.append(input[l])
    l = l +1
m = 4
while m <=74:
    y.append(temperature[m])
    m = m +1
y = np.matrix(y)
y = np.transpose(y)
phi_t= [a1, a2, b1, b2]
phi = np.transpose(phi_t)
phi = np.matrix(phi)
phi_t = np.matrix(phi_t)
pr_phi =  phi_t * phi
inv = np.linalg.inv(pr_phi)
inv = np.matrix(inv)
tita_pre = inv * phi_t
tita = tita_pre * y
u = [0]*70+ [1]*160+ [0]*80+ [1]*80+[0]
*40+[1]*40 + [0]*40 + [0]*40 + [1]*40 + [0]*20
u = u[0::4]
z = 3
resultd = []
time, temperature= np.loadtxt('digital_te.txt', unpack = True)
temperature = temperature[0::4]
while (z < 154):
```

```
56      y_est  = tita[0][0,0] * temperature[z + 22] + tita[1][0,0]
57      * temperature[z + 21]
58      + tita[2][0,0] * u[z-2]
59      + tita[3][0,0]*u[z-3]
60      resultd.append(y_est)
61      z = z + 1
62  resultd = np.resize(resultd,(1,150))
63  resultd = resultd.flatten() - 340.3963623046875
64  t = np.arange(1080,7080,40)
65  plt.plot(t,resultd, linewidth = 2, label = 'y estimada cada 40s')
66  leg = plt.legend(loc = 3)
```

Programa A.12: Modelo de indentificacion discreto con un tiempo de discretizacion de 10 sg de la T de agua a la salida respecto a la de entrada de humos

```
1  '''Calculo del modelo de indentificacion discreto con un tiempo
2  de discretizacion de 10 sg de la T de agua a la salida
3  respecto a los cambios de la de humos
4  a la entrada.
5  Representacion repecto a resultados experimentales'''
6  import numpy as np
7  import matplotlib.pyplot as plt
8  time, temperature = np.loadtxt('digital10_th.txt', unpack = True)
9  input  = [0]*50 + [-1]*50+ [0]*200
10 time = (time-4)*10
11 a1 = []
12 a2 = []
13 b1 = []
14 b2 = []
15 y = []
16 i = 9
17 while i < 299:
18     a1.append(temperature[i])
19     i = i +1
20 j = 8
21 while j < 298:
22     a2.append(temperature[j])
23     j = j +1
24 k = 7
25 while k < 297:
26     b1.append(input[k])
27     k = k+1
28 l = 6
29 while l < 296:
30     b2.append(input[l])
```

```
31      l = l +1
32  m = 10
33  while m <=299:
34      y.append(temperature[m])
35      m = m +1
36  y = np.matrix(y)
37  y = np.transpose(y)
38  phi_t= [a1, a2, b1, b2]
39  phi = np.transpose(phi_t)
40  phi = np.matrix(phi)
41  phi_t = np.matrix(phi_t)
42  pr_phi =  phi_t * phi
43  inv = np.linalg.inv(pr_phi)
44  inv = np.matrix(inv)
45  tita_pre = inv * phi_t
46  tita = tita_pre * y
47  u =[0]*60+  [-1]*50+[0]*200+ [-1]*50+[0]*50+[-1]*50 + [0]*150
48  z = 10
49  resultd = []
50  time, temperature= np.loadtxt('digital_th.txt', unpack = True)
51  while (z < 611):
52      y_est  = tita[0][0,0] * temperature[z + 188]
53      + tita[1][0,0] * temperature[z + 187]  + tita[2][0,0]
54      * u[z-3] + tita[3][0,0]*u[z-4]
55      resultd.append(y_est)
56      z = z + 1
57  resultd = np.resize(resultd,(1,600))
58  resultd = resultd.flatten() - 340.3963623046875
59  t = np.arange(2020,8020,10)
60  plt.plot(t,resultd, linewidth = 2, label = 'y estimada cada 10 s')
61  leg = plt.legend(loc = 3)
```

Programa A.13: Modelo de indentificacion discreto con un tiempo de discretizacion de 20 sg de la T de agua a la salida respecto a la de humos

```
1  '''Calculo del modelo de indentificacion discreto con un tiempo
2  de discretizacion de 20 sg de la T de agua a la salida
3  respecto a los cambios de la de humos
4  a la entrada.
5  Representacion repecto a resultados experimentales'''
6  import numpy as np
7  import matplotlib.pyplot as plt
8  time, temperature = np.loadtxt('digital10_th.txt', unpack = True)
9  input  = [0]*50 + [-1]*50+ [0]*200
10 time = (time-4)*10
```

```
11 temperature = temperature[0::2]
12 input = input[0::2]
13 a1 = []
14 a2 = []
15 b1 = []
16 b2 = []
17 y = []
18 i = 5
19 while i < 149:
20     a1.append(temperature[i])
21     i = i +1
22 j = 4
23 while j < 148:
24     a2.append(temperature[j])
25     j = j +1
26 k = 2
27 while k < 146:
28     b1.append(input[k])
29     k = k+1
30 l = 1
31 while l < 145:
32     b2.append(input[l])
33     l = l +1
34 m = 6
35 while m <=149:
36     y.append(temperature[m])
37     m = m +1
38 y = np.matrix(y)
39 y = np.transpose(y)
40 phi_t= [a1, a2, b1, b2]
41 phi = np.transpose(phi_t)
42 phi = np.matrix(phi)
43 phi_t = np.matrix(phi_t)
44 pr_phi =  phi_t * phi
45 inv = np.linalg.inv(pr_phi)
46 inv = np.matrix(inv)
47 tita_pre = inv * phi_t
48 tita = tita_pre * y
49 u =[0]*60+  [-1]*50+[0]*200+ [-1]*50+[0]*50+[-1]*50 + [0]*150
50 u = u[0::2]
51 z = 5
52 resultd = []
53 time, temperature= np.loadtxt('digital_th.txt', unpack = True)
54 temperature = temperature[0::2]
```

```
55 while (z < 305):
56     y_est  = tita[0][0,0] * temperature[z + 94] + tita[1][0,0]
57     * temperature[z + 93]  + tita[2][0,0] * u[z-2] + tita[3][0,0]*u[z-3]
58     resultd.append(y_est)
59     z = z + 1
60 resultd = np.resize(resultd,(1,300))
61 resultd = resultd.flatten() - 340.3963623046875
62 t = np.arange(2040,8040,20)
63 plt.plot(t,resultd, linewidth = 2, label = 'y estimada cada 20 s')
64 leg = plt.legend(loc = 3)
```

Programa A.14: Modelo de indentificacion discreto con un tiempo de discretizacion de 40 sg de la T de agua a la salida respecto a la de humos

```
1  '''Calculo del modelo de indentificacion discreto con un tiempo
2  de discretizacion de 40 sg de la T de agua a la salida
3  respecto a los cambios de la de humos
4  a la entrada.
5  Representacion repecto a resultados experimentales'''
6  import numpy as np
7  import matplotlib.pyplot as plt
8  time, temperature = np.loadtxt('digital10_th.txt', unpack = True)
9  input  = [0]*50 + [-1]*50+ [0]*200
10 time = (time-4)*10
11 temperature = temperature[0::2]
12 input = input[0::4]
13 a1 = []
14 a2 = []
15 b1 = []
16 b2 = []
17 y = []
18 i = 3
19 while i < 74:
20     a1.append(temperature[i])
21     i = i +1
22 j = 2
23 while j < 73:
24     a2.append(temperature[j])
25     j = j +1
26 k = 3
27 while k < 74:
28     b1.append(input[k])
29     k = k+1
30 l = 2
31 while l < 73:
```

```
32      b2.append(input[l])
33      l = l +1
34  m = 4
35  while m <=74:
36      y.append(temperature[m])
37      m = m +1
38  y = np.matrix(y)
39  y = np.transpose(y)
40  phi_t= [a1, a2, b1, b2]
41  phi = np.transpose(phi_t)
42  phi = np.matrix(phi)
43  phi_t = np.matrix(phi_t)
44  pr_phi =  phi_t * phi
45  inv = np.linalg.inv(pr_phi)
46  inv = np.matrix(inv)
47  tita_pre = inv * phi_t
48  tita = tita_pre * y
49  u =[0]*60+  [-1]*50+[0]*200+ [-1]*50+[0]*50+[-1]*50 + [0]*150
50  u = u[0::4]
51  z = 2
52  resultd = []
53  time, temperature= np.loadtxt('digital_th.txt', unpack = True)
54  temperature = temperature[0::4]
55  while (z < 153):
56      y_est  = tita[0][0,0] * temperature[z + 47] + tita[1][0,0]
57      * temperature[z + 46]  + tita[2][0,0] * u[z-1]
58      + tita[3][0,0]*u[z-2]
59      resultd.append(y_est)
60      z = z + 1
61  resultd = np.resize(resultd,(1,150))
62  resultd = resultd.flatten() - 340.3963623046875
63  t = np.arange(2040,8040,40)
64  plt.plot(t,resultd, linewidth = 2, label = 'y estimada cada 40 s')
65  leg = plt.legend(loc = 3)
```