Grupo 3 - Proyecto Grupal

July 7, 2021

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import scipy.optimize as sopt
import scipy.stats as stats
from scipy.stats import anderson
```

[2]:
```
datos=pd.read_excel('Datos.xlsx')
```

[3]:
```
datos=datos.dropna()
```

[4]:
```
datos.head()
```

[4]:

	Datos destilación	Unnamed: 1	Unnamed: 2
2	Volumen evaporado, %	tiempo, s	Temperatura de destilación, °C
3	Primera gota	375	26.6
4	5	460	43.5
5	10	529.7	50.2
6	15	599.5	55.7

	Unnamed: 3	Unnamed: 4	Unnamed: 5
2	Volumen evaporado, %	tiempo	Temperatura de destilación
3	Primera gota	376	29.6
4	5	458	45.8
5	10	529.4	52.6
6	15	599.1	58

	Unnamed: 6	Unnamed: 7	Unnamed: 8
2	Volumen evaporado, %	tiempo	Temperatura de destilación
3	Primera gota	379	28.4
4	5	459	44.3
5	10	530.4	51
6	15	601.8	56.8

	Unnamed: 9	Unnamed: 10	Unnamed: 11
2	Volumen evaporado, %	tiempo	Temperatura de destilación
3	Primera gota	375	30.8
4	5	465	47.4

```
5                    10       536.4                          52.3
6                    15       607.8                          58.3
```

[5]:
```
datosCorrida1=pd.concat([datos['Datos destilación'],datos['Unnamed:␣
↪1'],datos['Unnamed: 2']],axis=1)
datosCorrida2=pd.concat([datos['Unnamed: 3'],datos['Unnamed: 4'],datos['Unnamed:
↪ 5']],axis=1)
datosCorrida3=pd.concat([datos['Unnamed: 6'],datos['Unnamed: 7'],datos['Unnamed:
↪ 8']],axis=1)
datosCorrida4=pd.concat([datos['Unnamed: 9'],datos['Unnamed:␣
↪10'],datos['Unnamed: 11']],axis=1)
```

[6]:
```
datosCorrida2.head()
```

[6]:
```
             Unnamed: 3 Unnamed: 4                  Unnamed: 5
2  Volumen evaporado, %     tiempo  Temperatura de destilación
3          Primera gota        376                        29.6
4                     5        458                        45.8
5                    10      529.4                        52.6
6                    15      599.1                          58
```

[7]:
```
datosCorrida1.columns=datosCorrida1.iloc[0]
datosCorrida2.columns=datosCorrida2.iloc[0]
datosCorrida3.columns=datosCorrida3.iloc[0]
datosCorrida4.columns=datosCorrida4.iloc[0]
```

[8]:
```
datosCorrida1=datosCorrida1.drop(2,axis=0)
datosCorrida2=datosCorrida2.drop(2,axis=0)
datosCorrida3=datosCorrida3.drop(2,axis=0)
datosCorrida4=datosCorrida4.drop(2,axis=0)
```

[9]:
```
datosCorrida1.head()
```

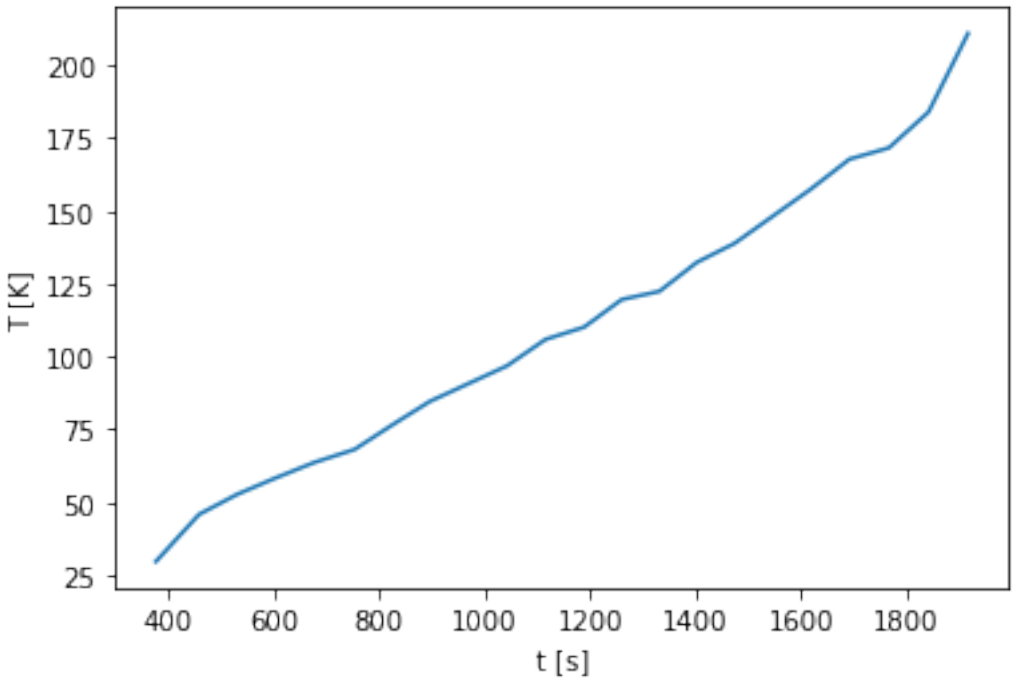
[9]:
```
2 Volumen evaporado, % tiempo, s Temperatura de destilación, °C
3         Primera gota       375                          26.6
4                    5       460                          43.5
5                   10     529.7                          50.2
6                   15     599.5                          55.7
7                   20     674.5                          61.4
```

[10]:
```
datosCorrida1.index=[i for i in range(len(datosCorrida1))]
datosCorrida2.index=[i for i in range(len(datosCorrida2))]
datosCorrida3.index=[i for i in range(len(datosCorrida3))]
datosCorrida4.index=[i for i in range(len(datosCorrida4))]
```

[11]:
```
datosCorrida2.head()
```

[11]:
	2 Volumen evaporado, %	tiempo	Temperatura de destilación
0	Primera gota	376	29.6
1	5	458	45.8
2	10	529.4	52.6
3	15	599.1	58
4	20	676.1	63.6

[12]:
```
plt.plot(datosCorrida2['tiempo'],datosCorrida2['Temperatura de destilación'])
plt.xlabel('t [s]')
plt.ylabel('T [K]')
plt.show()
```

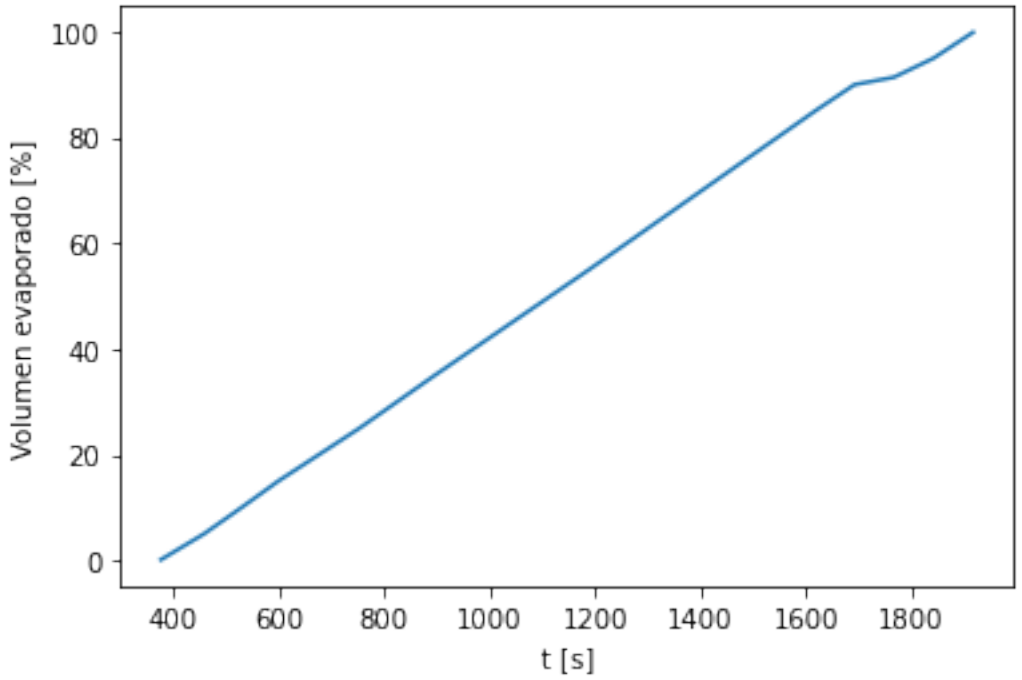


[13]:
```
datosCorrida1['Volumen evaporado, %'][0]=0.1
datosCorrida2['Volumen evaporado, %'][0]=0.1
datosCorrida3['Volumen evaporado, %'][0]=0.1
datosCorrida4['Volumen evaporado, %'][0]=0.1
```

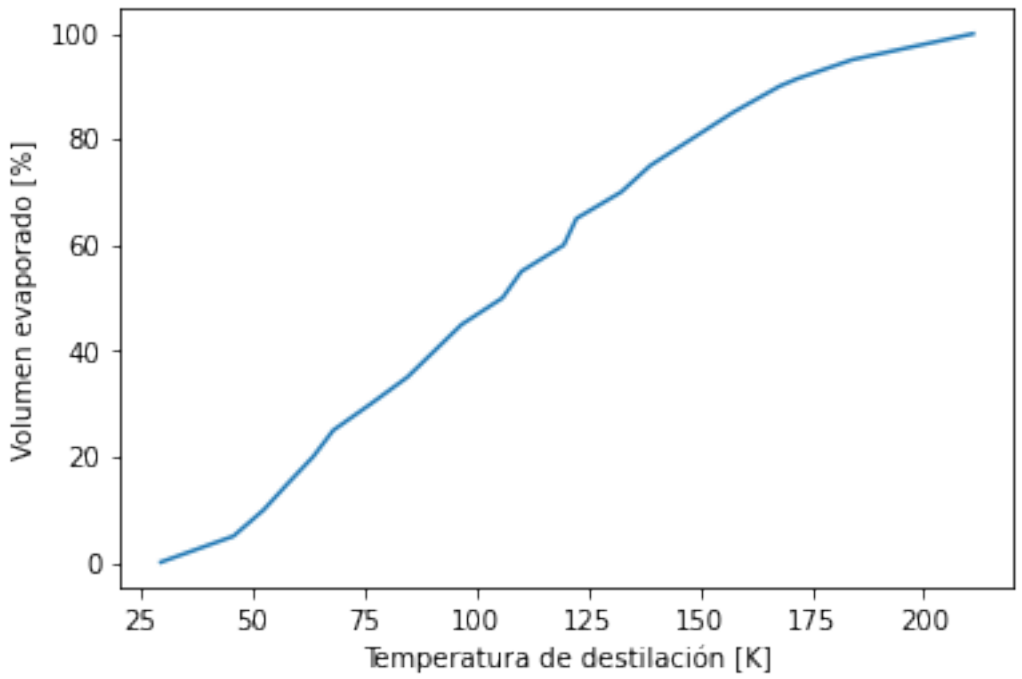
[14]:
```
datosCorrida3.head()
```

[14]:
	2 Volumen evaporado, %	tiempo	Temperatura de destilación
0	0.1	379	28.4
1	5	459	44.3
2	10	530.4	51
3	15	601.8	56.8
4	20	676.8	62.3

[15]:
```
plt.plot(datosCorrida2['tiempo'],datosCorrida2['Volumen evaporado, %'])
plt.xlabel('t [s]')
plt.ylabel('Volumen evaporado [%]')
plt.show()
```



[16]:
```
plt.plot(datosCorrida2['Temperatura de destilación'],datosCorrida2['Volumen␣
↪evaporado, %'])
plt.xlabel('Temperatura de destilación [K]')
plt.ylabel('Volumen evaporado [%]')
plt.show()
```



1 Modelo empírico polinomial

Para la temperatura como función del tiempo, se observa que existe cierta linealidad; sin embargo, podría expandirse un polinomio de grado mayor, lo que permitiría aumentar el ajuste, de esta forma, se escoge un polinomio de ajuste de la forma:

$$T(t) = At^3 + Bt^2 + Ct + D$$

[17]:
```
volumen,tiempo,temperatura=np.asarray(datosCorrida2['Volumen evaporado, %']).
↪astype('float64'),np.asarray(datosCorrida2['tiempo']).astype('float64'),np.
↪asarray(datosCorrida2['Temperatura de destilación']+273.15).astype('float64')
```

[18]:
```
sd2=(temperatura-temperatura.mean()).sum()**2/(1*(len(temperatura)-1))
sd2
```

[18]:
```
9.8473882446789e-27
```

[19]:
```
def TPolinomio(t,A,B,C,D):
    return A*t**3+B*t**2+C*t+D
```

[20]:
```
coefsP,covP=sopt.curve_fit(TPolinomio,tiempo,temperatura)
ajusTPoli=TPolinomio(tiempo,coefsP[0],coefsP[1],coefsP[2],coefsP[3])
residuosEmpíricoPoli=ajusTPoli-temperatura
```

[21]:
```
residuosEmpíricoPoli.std()
```

[21]: 3.029611693781791

[22]:
```
coefsP
```

[22]: array([3.75542223e-08, -1.09570116e-04, 1.91978498e-01, 2.46747689e+02])

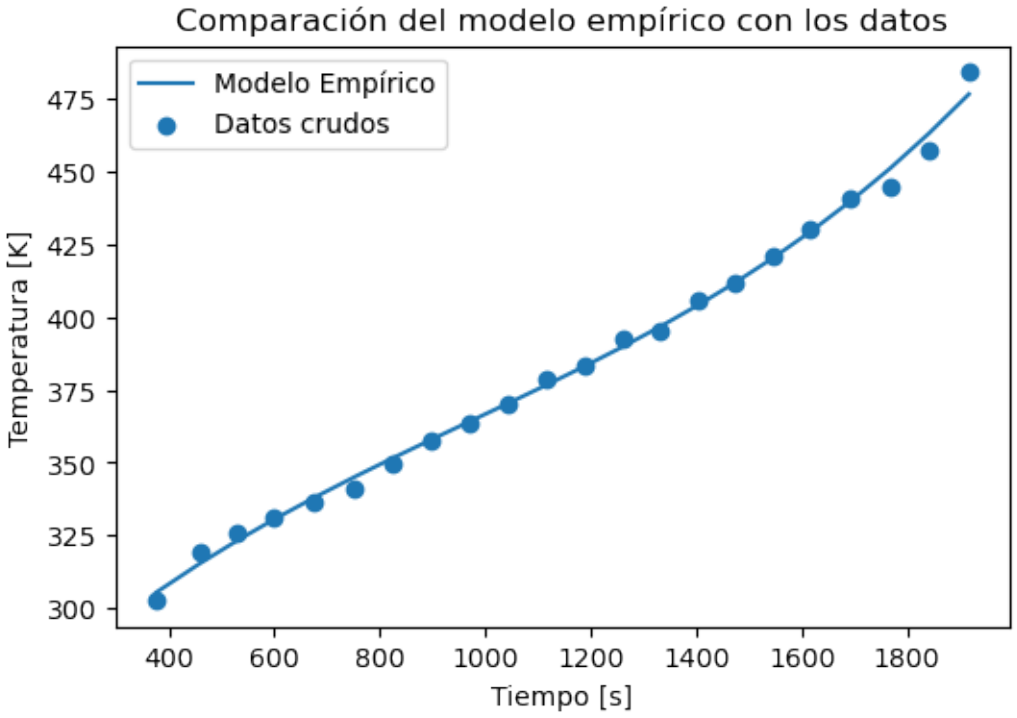
[23]:
```
se2P=(residuosEmpíricoPoli**2).sum()/(len(residuosEmpíricoPoli)-4)
np.sqrt(se2P)
```

[23]: 3.3493617495824335

[24]:
```
sm2P=se2P-sd2
np.sqrt(sm2P)
```

[24]: 3.3493617495824335

[25]:
```
plt.figure(dpi=100)
plt.plot(tiempo,ajusTPoli)
plt.scatter(tiempo,temperatura)
plt.title('Comparación del modelo empírico con los datos')
plt.xlabel('Tiempo [s]')
plt.ylabel('Temperatura [K]')
plt.legend(['Modelo Empírico','Datos crudos'])
plt.savefig('ModeloEmpírico.png')
plt.show()
```



1.0.1 Diagnósticos de los residuos para el modelo polinómico

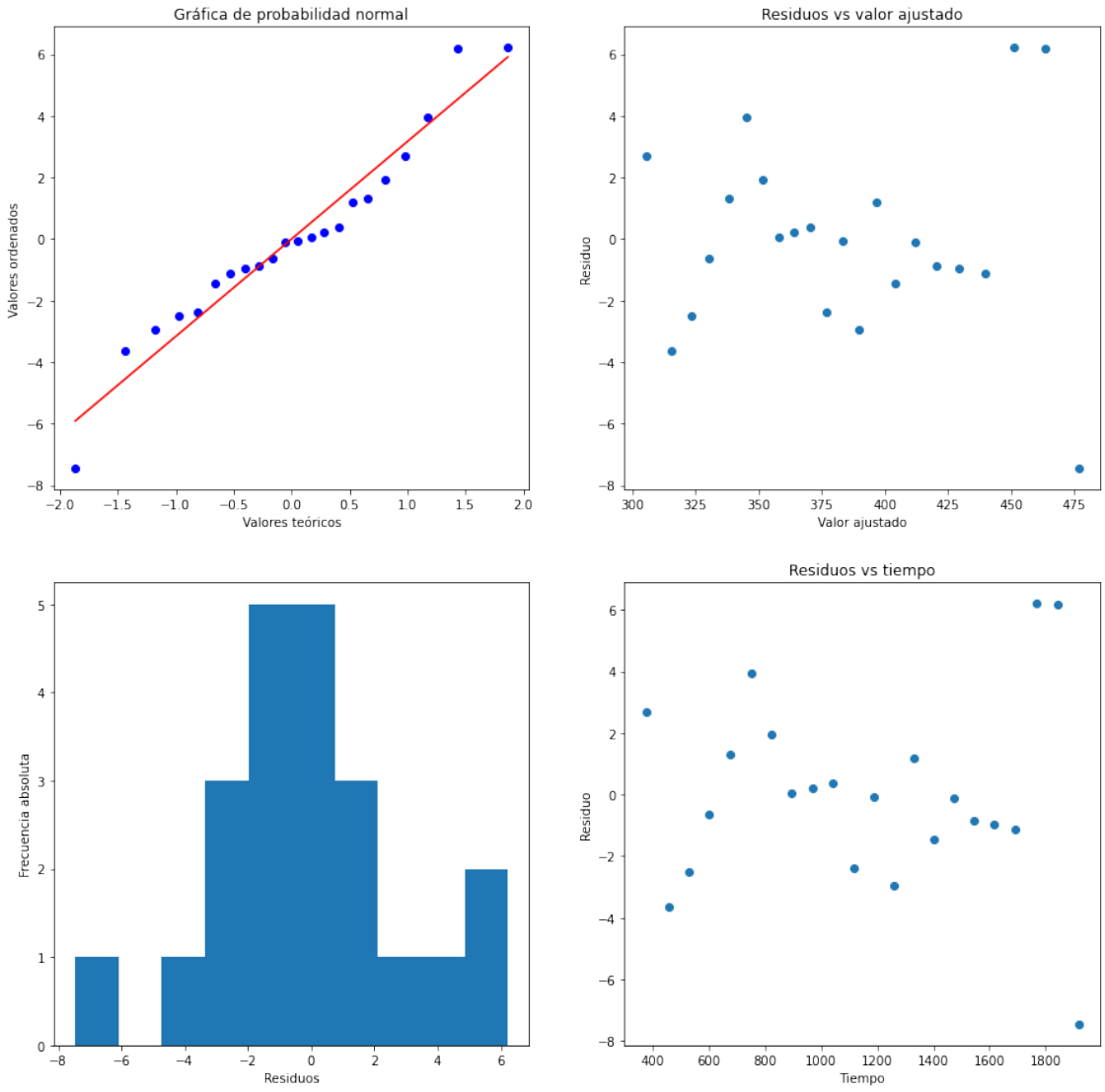
Prueba de normalidad Se observa que el coeficiente de Anderson-Darling calculado es menor al valor teórico con un 95% de confianza, por lo que se acepta la normalidad de los residuos.

[26]:
```
anderson(residuosEmpíricoPoli).statistic,anderson(residuosEmpíricoPoli).
↪critical_values[2]
```

[26]:
```
(0.41777808179304543, 0.696)
```

Gráficas de residuos

[27]:
```
fig1,((ax1,ax2),(ax3,ax4))=plt.subplots(2,2,figsize=(15,15))
stats.probplot(residuosEmpíricoPoli,dist='norm',plot=ax1)
ax1.set_title('Gráfica de probabilidad normal')
ax1.set_xlabel('Valores teóricos')
ax1.set_ylabel('Valores ordenados')
ax2.scatter(ajusTPoli,residuosEmpíricoPoli)
ax2.set_title('Residuos vs valor ajustado')
ax2.set_ylabel('Residuo')
ax2.set_xlabel('Valor ajustado')
ax3.hist(residuosEmpíricoPoli)
ax3.set_xlabel('Residuos')
ax3.set_ylabel('Frecuencia absoluta')
ax4.scatter(tiempo,residuosEmpíricoPoli)
ax4.set_title('Residuos vs tiempo')
ax4.set_ylabel('Residuo')
ax4.set_xlabel('Tiempo')
fig1.suptitle('Diagnósticos de los residuos del modelo empírico polinómico')
plt.savefig('DiagnósticoEmpírico.png')
plt.show()
```



2 Modelo teórico

2.1 Balance de energía

Se define como volumen de control la muestra de hidrocarburos, se tiene entonces que:

$$V_A = F_E - F_S + V_G - V_C$$

- Velocidad de acumulación, $V_A = \frac{dM_e}{dt}$
- Flujo de entrada, $F_E = \dot{Q}$

- Flujo de salida, $F_S = \dot{M}_S \Delta H$
- Velocidad de generación, $R_P = 0$
- Velocidad de consumo, $R_C = 0$

$$\frac{dM_e}{dt} = \dot{Q} - \dot{M}_S \Delta H$$
$$\Rightarrow \frac{dMC\,(T - T_{ref})}{dt} = \dot{Q} - \dot{M}_S \Delta H$$
$$\Rightarrow C\frac{dM\,(T - T_{ref})}{dt} = \dot{Q} - \dot{M}_S \Delta H$$
$$\Rightarrow C\left((T - T_{ref})\,\frac{dM}{dt} + \frac{dT}{dt}M\right) = \dot{Q} - \rho\dot{V}_S \Delta H$$

2.1.1 Balance de masa total

$$\frac{dM}{dt} = -\dot{M}_s$$

Integrando:

$$M = C_1 - \dot{M}_S t$$

Con $C_1 = M_0 = \rho_L V_0$ la masa inicial en el volumen de control.

Por otro lado $\dot{M}_S = \rho_V \dot{V}_S$

2.1.2 Sustitución

$$\Rightarrow C\left(-\dot{M}_S\,(T - T_{ref}) + \frac{dT}{dt}\left(M_0 - \dot{M}_S t\right)\right) = \dot{Q} - \dot{M}_S \Delta H$$

Tomando $T_{ref} = 0$ K $= -273.15$ °C para la temperatura de referencia de la capacidad calorífica:

$$\Rightarrow C\left(-\dot{M}_S\,(T - T_{ref}) + \frac{dT}{dt}\left(M_0 - \dot{M}_S t\right)\right) = \dot{Q} - \dot{M}_S \Delta H$$

Simplificando:

$$\frac{dT}{dt} - \frac{\dot{M}_S}{M_0 - \dot{M}_S t}T = \frac{\dot{Q} - \dot{M}_S \Delta H}{C\left(M_0 - \dot{M}_S t\right)} - \frac{\dot{M}_S}{M_0 - \dot{M}_S t}T_{ref}$$

Tomando $\Delta H(t) = a + bt$:

$$\frac{dT}{dt} - \frac{\dot{M}_S}{M_0 - \dot{V}_S t}T = \frac{\dot{Q} - \dot{M}_S a - \dot{M}_S bt}{C\left(M_0 - \dot{M}_S t\right)} - \frac{\dot{M}_S}{M_0 - \dot{M}_S t}T_{ref}$$

Ahora se debe buscar la solución de esta expresión, para esto, se toma el factor integrante:

$$F_I(t) = \exp\left(\int_0^t -\frac{\dot{M}_S}{M_0 - \dot{M}_S t'} dt'\right) = M_0 - \dot{M}_S t$$

Multilicando a ambos lados la ecuación por el factor integrante:

$$\left(M_0 - \dot{M}_S t\right)\frac{dT}{dt} - \dot{M}_S T = \frac{\dot{Q} - \dot{M}_S a - \dot{M}_S bt}{C} - \dot{M}_S T_{ref}$$

Esto se puede simplificar de la forma:

$$\frac{d}{dt}\left(\left(M_0 - \dot{M}_S t\right)T\right) = \frac{\dot{Q} - \dot{M}_S a - \dot{M}_S bt}{C} - \dot{M}_S T_{ref}$$

Integrando:

$$\left(M_0 - \dot{M}_S t\right)T = \frac{\left(2\dot{Q} - 2\dot{M}_S a\right)t - \dot{M}_S bt^2}{2C} - \dot{M}_S T_{ref} t + C_2$$

$$\Rightarrow T = \frac{\left(2\dot{Q} - 2\dot{M}_S a\right)t - \dot{M}_S bt^2}{2C\left(M_0 - \dot{M}_S t\right)} + \frac{C_2 - \dot{M}_S T_{ref} t}{\left(M_0 - \dot{M}_S t\right)}$$

En $t = t_i$, $T = T_i$, entonces:

$$T_i = \frac{2\left(\dot{Q} - \dot{M}_S a\right)t_i - \dot{M}_S bt_i^2}{2CM_0 - 2C\dot{M}_S t_i} + \frac{C_2 - \dot{M}_S T_{ref} t_i}{\left(M_0 - \dot{M}_S t_i\right)}$$

$$\Rightarrow T_i\left(M_0 - \dot{M}_S t_i\right) - \frac{2\left(\dot{Q} - \dot{M}_S a\right)t_i - \dot{M}_S bt_i^2}{2C} + \dot{M}_S T_{ref} t_i = C_2$$

$$C_2 = \frac{2CT_i\left(M_0 - \dot{M}_S t_i\right) - 2\left(\dot{Q} - \dot{M}_S a\right)t_i + \dot{M}_S bt_i^2 + 2C\dot{M}_S T_{ref} t_i}{2C}$$

Sustituyendo:

Si se trabaja en Celsius:

$$T = \frac{2\left(\dot{Q} - \dot{M}_S a\right)t - \dot{M}_S bt^2}{2C\left(M_0 - \dot{M}_S t\right)} + \frac{2CT_i\left(M_0 - \dot{M}_S t_i\right) - 2\left(\dot{Q} - \dot{M}_S a\right)t_i + \dot{M}_S bt_i^2 + 2C\dot{M}_S T_{ref} t_i - 2C\dot{M}_S T_{ref} t}{2C\left(M_0 - \dot{M}_S t\right)}$$

Pero en Kelvin:

$$T = \frac{2\left(\dot{Q} - \dot{M}_S a\right)t - \dot{M}_S bt^2}{2C\left(M_0 - \dot{M}_S t\right)} + \frac{2CT_i\left(M_0 - \dot{M}_S t_i\right) - 2\left(\dot{Q} - \dot{M}_S a\right)t_i + \dot{M}_S bt_i^2}{2C\left(M_0 - \dot{M}_S t\right)}$$

$$T = \frac{\alpha t - \beta t^2 + \gamma}{\left(M_0 - \dot{M}_S t\right)}$$

[28]:
```
def Teórico(t,alpha,beta,gamma,M0,Ms):
    return (alpha*t-beta*t**2+gamma)/(M0-Ms*t)
```

[29]:
```
coefsT,covT=sopt.curve_fit(Teórico,tiempo,temperatura)
ajusT=Teórico(tiempo,coefsT[0],coefsT[1],coefsT[2],coefsT[3],coefsT[4])
residuosTeórico=ajusT-temperatura
```

[30]:
```
residuosTeórico.std()
```

[30]:
```
2.2112840737912043
```

[31]:
```
coefsT
```

[31]:
```
array([ 1.64996382e+03, -1.77328215e+00, -1.02491714e+07, -3.75993309e+04,
       -1.89996118e+01])
```

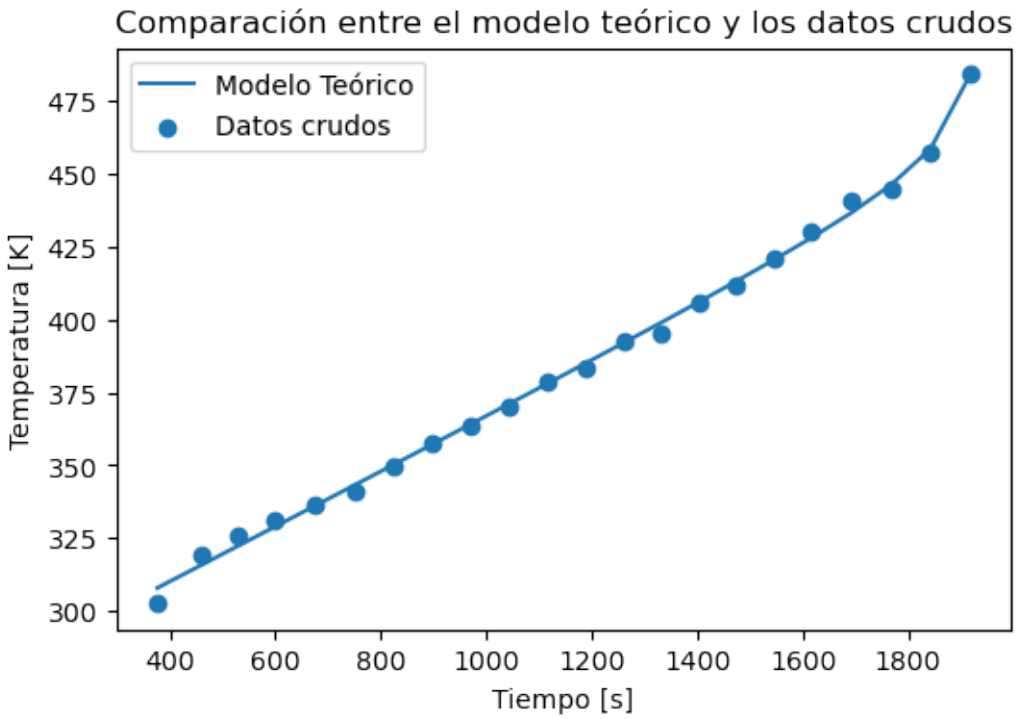
[32]:
```
se2T=(residuosTeórico**2).sum()/(len(residuosTeórico)-5)
np.sqrt(se2T)
```

[32]:
```
2.5155411020176692
```

[33]:
```
sm2T=se2T-sd2
np.sqrt(sm2T)
```

[33]:
```
2.5155411020176692
```

[34]:
```
plt.figure(dpi=100)
plt.plot(tiempo,ajusT)
plt.scatter(tiempo,temperatura)
plt.legend(['Modelo Teórico','Datos crudos'])
plt.title('Comparación entre el modelo teórico y los datos crudos')
plt.xlabel('Tiempo [s]')
plt.ylabel('Temperatura [K]')
plt.savefig('ModeloTeórico.png')
plt.show()
```

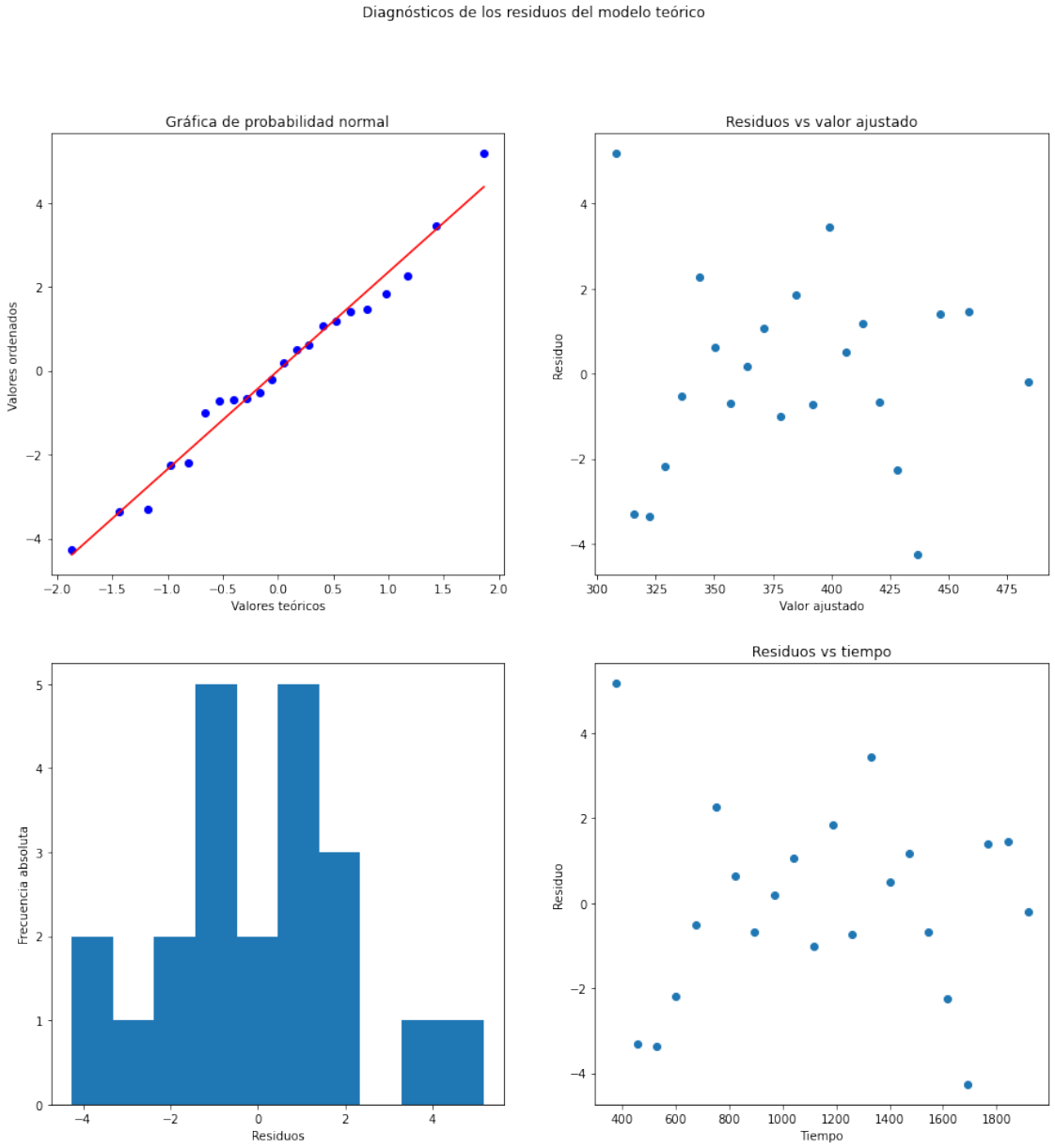


[35]:
```
anderson(residuosTeórico,dist='norm').
↪statistic,anderson(residuosTeórico,dist='norm').critical_values
```

[35]:
```
(0.20604843235782866, array([0.51 , 0.58 , 0.696, 0.812, 0.966]))
```

[36]:
```
fig4,((ax1,ax2),(ax3,ax4))=plt.subplots(2,2,figsize=(15,15))
stats.probplot(residuosTeórico,dist='norm',plot=ax1)
ax1.set_title('Gráfica de probabilidad normal')
ax1.set_xlabel('Valores teóricos')
ax1.set_ylabel('Valores ordenados')
ax2.scatter(ajusT,residuosTeórico)
ax2.set_title('Residuos vs valor ajustado')
ax2.set_ylabel('Residuo')
ax2.set_xlabel('Valor ajustado')
ax3.hist(residuosTeórico)
ax3.set_xlabel('Residuos')
ax3.set_ylabel('Frecuencia absoluta')
ax4.scatter(tiempo,residuosTeórico)
ax4.set_title('Residuos vs tiempo')
ax4.set_ylabel('Residuo')
ax4.set_xlabel('Tiempo')
fig4.suptitle('Diagnósticos de los residuos del modelo teórico')
plt.savefig('DiagnósticoTeórico.png')
```

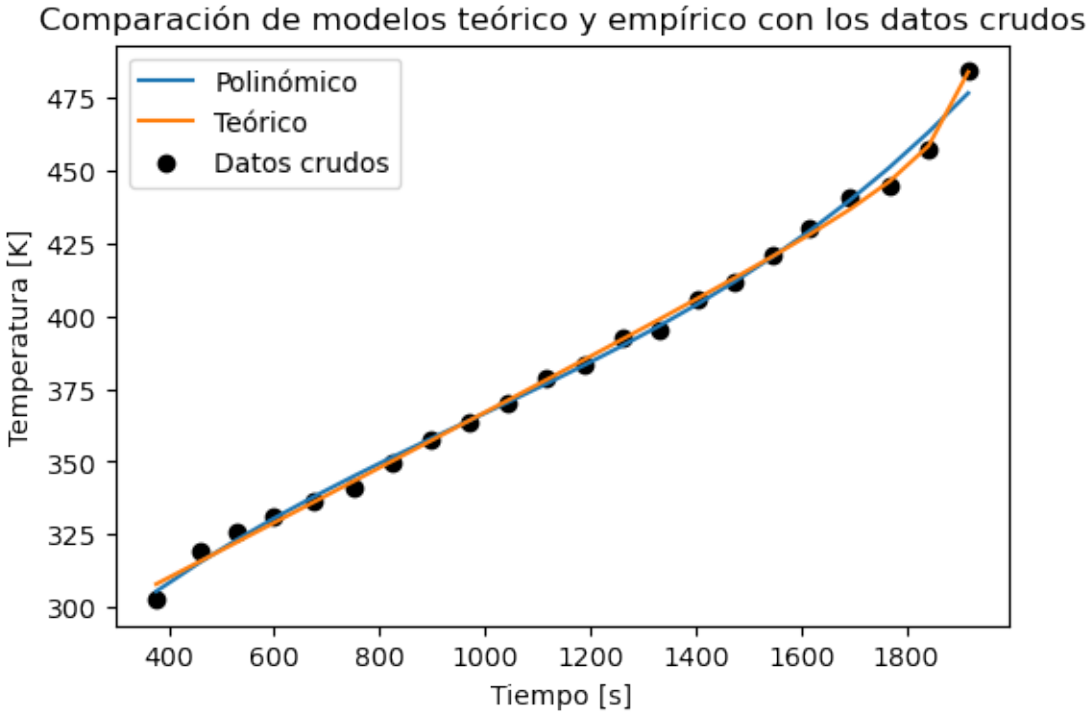
```
plt.show()
```



3 Comparación de los modelos

El estadístico F a 16 grados de libertad para numerador y 18 para el denominador denominador tiene un valor teórico de 2.25 al 95% y un valor de 3.190 al 99%, comparando con el calculado, se determina que las varianzas de los residuos para ambos modelos son distintas.

[37]:
```
plt.figure(dpi=100)
plt.scatter(tiempo,temperatura,color='black')
plt.plot(tiempo,ajusTPoli)
plt.plot(tiempo,ajusT)
plt.legend(['Polinómico','Teórico','Datos crudos'])
plt.title('Comparación de modelos teórico y empírico con los datos crudos')
plt.xlabel('Tiempo [s]')
plt.ylabel('Temperatura [K]')
plt.savefig('AmbosModelos.png')
plt.show()
```



3.1 Desviaciones de la estima

[38]:
```
F_calc=se2P/se2T
F_calc
```

[38]: 1.7728062617861313

[39]:
```
1-stats.f.cdf(F_calc,22-4,22-5)
```

[39]: 0.12208143982554465

Se puede concluir que los modelos son estadísticamente iguales al 95% y 99% de confianza.

3.2 Desviación de los residuos

Al 95% el valor del estadístico F teórico es 2.084, por lo que se observa que para este nivel de confianza las variabilidades de los residuos entre ambos modelos son iguales; además, al 99% de confianza el estadístico F teórico es 2.857, y entonces se puede concluir que las desviaciones de los residuos de ambos modelos son estadísticamente iguales al 95% y al 99% de confianza

[40]:
```
residuosEmpíricoPoli.std()**2/residuosTeórico.std()**2
```

[40]:
```
1.8770889830838418
```

[41]:
```
1-stats.f.cdf(residuosEmpíricoPoli.std()**2/residuosTeórico.
↪std()**2,len(residuosTeórico)-1,len(residuosEmpíricoPoli)-1)
```

[41]:
```
0.07861896296997561
```

3.3 Comparación de la varianza de los residuos con una arbitraria

3.3.1 Para el teórico

A 17 grados de libertad el χ^2 teórico tiene un valor de 26.296, para un 95% de confianza, como se obtiene un número menor a este, se puede rechazar la hipótesis nula de que la varianza de los residuos es igual a la varianza $\sigma_0^2 = (5)^2$, al 95% de confianza.

Al 99% de confianza, el valor de χ^2 es 32.000, de forma que se puede afirmar que las varianzas de los residuos de los modelos es distinta al 99% de confianza.

3.3.2 Para el empírico

A 18 grados de libertad el χ^2 teórico tiene un valor de 28.869, para un 95% de confianza, como se obtiene un número menor a este, se puede aceptar la hipótesis nula de que la varianza de los residuos es distinta a la varianza $\sigma_0^2 = (5)^2$, al 95% de confianza.

Al 99% de confianza, el valor de χ^2 es 34.805, de forma que se puede afirmar que las varianzas de los residuos de los modelos es distinta al 99% de confianza.

[42]:
```
(22-5),(22-4)
```

[42]:
```
(17, 18)
```

[43]:
```
residuosTeórico.std()**2*(22-5)/5**2
```

[43]:
```
3.3250485334017843
```

[44]:
```
residuosEmpíricoPoli.std()**2*(22-4)/5**2
```

[44]:
```
6.608553850871549
```

4 Ajustando para todas las corridas experimentales

[45]:
```
tiempo1,temp1=np.asarray(datosCorrida1['tiempo, s']).astype('float64'),np.
 ↪asarray(datosCorrida1['Temperatura de destilación, °C']+273.15).
 ↪astype('float64')
tiempo2,temp2=tiempo.copy(),temperatura.copy()
tiempo3,temp3=np.asarray(datosCorrida3['tiempo']).astype('float64'),np.
 ↪asarray(datosCorrida3['Temperatura de destilación']+273.15).astype('float64')
tiempo4,temp4=np.asarray(datosCorrida4['tiempo']).astype('float64'),np.
 ↪asarray(datosCorrida4['Temperatura de destilación']+273.15).astype('float64')
```

[46]:
```
sd2=((temp1-temp1.mean()).sum()**2+(temp3-temp3.mean()).sum()**2+(temp4-temp4.
 ↪mean()).sum()**2)/(3*(len(temp1)-1))
sd2
```

[46]: 2.7388048555513193e-26

[47]:
```
volumen1=np.asarray(datosCorrida1['Volumen evaporado, %']).astype('float64')
volumen2=volumen.copy()
volumen3=np.asarray(datosCorrida3['Volumen evaporado, %']).astype('float64')
volumen4=np.asarray(datosCorrida4['Volumen evaporado, %']).astype('float64')
```

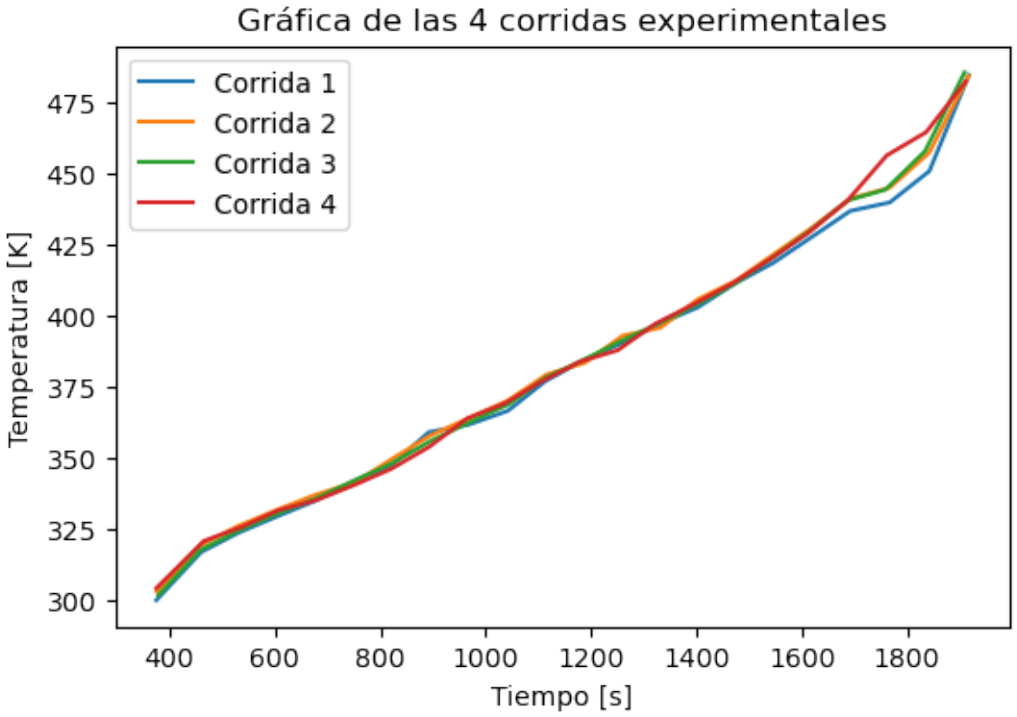
[48]:
```
tiempo=np.concatenate((tiempo1,tiempo2,tiempo4))
temperatura=np.concatenate((temp1,temp2,temp4))
volumen=np.concatenate((volumen1,volumen2,volumen4))
corridas=[[volumen[i],tiempo[i],temperatura[i]]for i in range(len(tiempo))]
corridas.sort(key = lambda x:x[1])
```

[49]:
```
corridas=np.array(corridas)
```

[50]:
```
corrida_avg=[]
for i in range(int(len(corridas)/4)):
    j=i*3
    k=j+3
    corrida_avg.append([corridas[j:k,0].mean(),corridas[j:k,1].
 ↪mean(),corridas[j:k,2].mean()])
corrida_avg=np.array(corrida_avg)
```

[51]:
```
plt.figure(dpi=100)
plt.plot(tiempo1,temp1)
plt.plot(tiempo2,temp2)
plt.plot(tiempo3,temp3)
plt.plot(tiempo4,temp4)
plt.xlabel('Tiempo [s]')
plt.ylabel('Temperatura [K]')
plt.legend(['Corrida 1','Corrida 2','Corrida 3','Corrida 4'])
plt.title('Gráfica de las 4 corridas experimentales')
```

```
plt.savefig('CorridasJuntas.png')
plt.show()
```



4.1 Modelo teórico

[52]:
```
#coefsT,_=sopt.curve_fit(Teórico,corridas[:,1],corridas[:,2])
ajusT=Teórico(corridas[:,1],coefsT[0],coefsT[1],coefsT[2],coefsT[3],coefsT[4])
residuosTeórico=ajusT-corridas[:,2]
```

[53]:
```
coefsT
```

[53]:
```
array([ 1.64996382e+03, -1.77328215e+00, -1.02491714e+07, -3.75993309e+04,
       -1.89996118e+01])
```

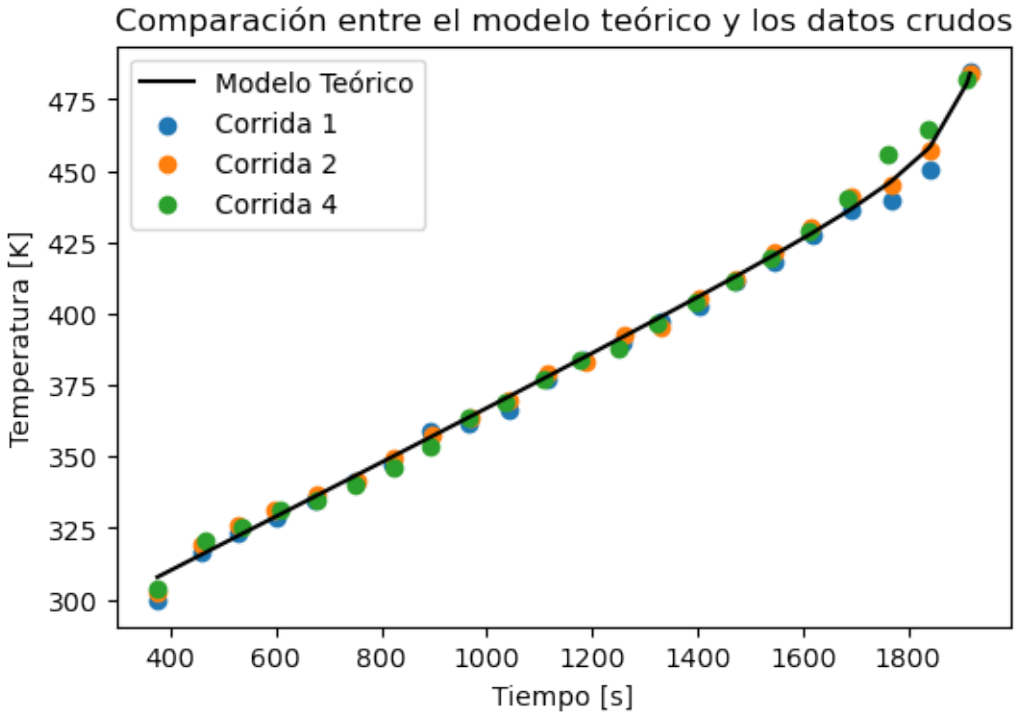
[54]:
```
se2T=(residuosTeórico**2).sum()/(3*len(residuosTeórico)-5)
np.sqrt(se2T)
```

[54]:
```
1.821004493455187
```

[55]:
```
sm2T=se2T-sd2
np.sqrt(sm2T)
```

[55]:
```
1.821004493455187
```

[56]:
```
plt.figure(dpi=100)
plt.plot(corridas[:,1],ajusT,color='black')
plt.scatter(tiempo1,temp1)
plt.scatter(tiempo2,temp2)
#plt.scatter(tiempo3,temp3)
plt.scatter(tiempo4,temp4)
plt.xlabel('Tiempo [s]')
plt.ylabel('Temperatura [K]')
plt.legend(['Modelo Teórico','Corrida 1','Corrida 2','Corrida 4'])
plt.title('Comparación entre el modelo teórico y los datos crudos')
plt.savefig('ModeloTeóricovsCorridas.png')
plt.show()
```

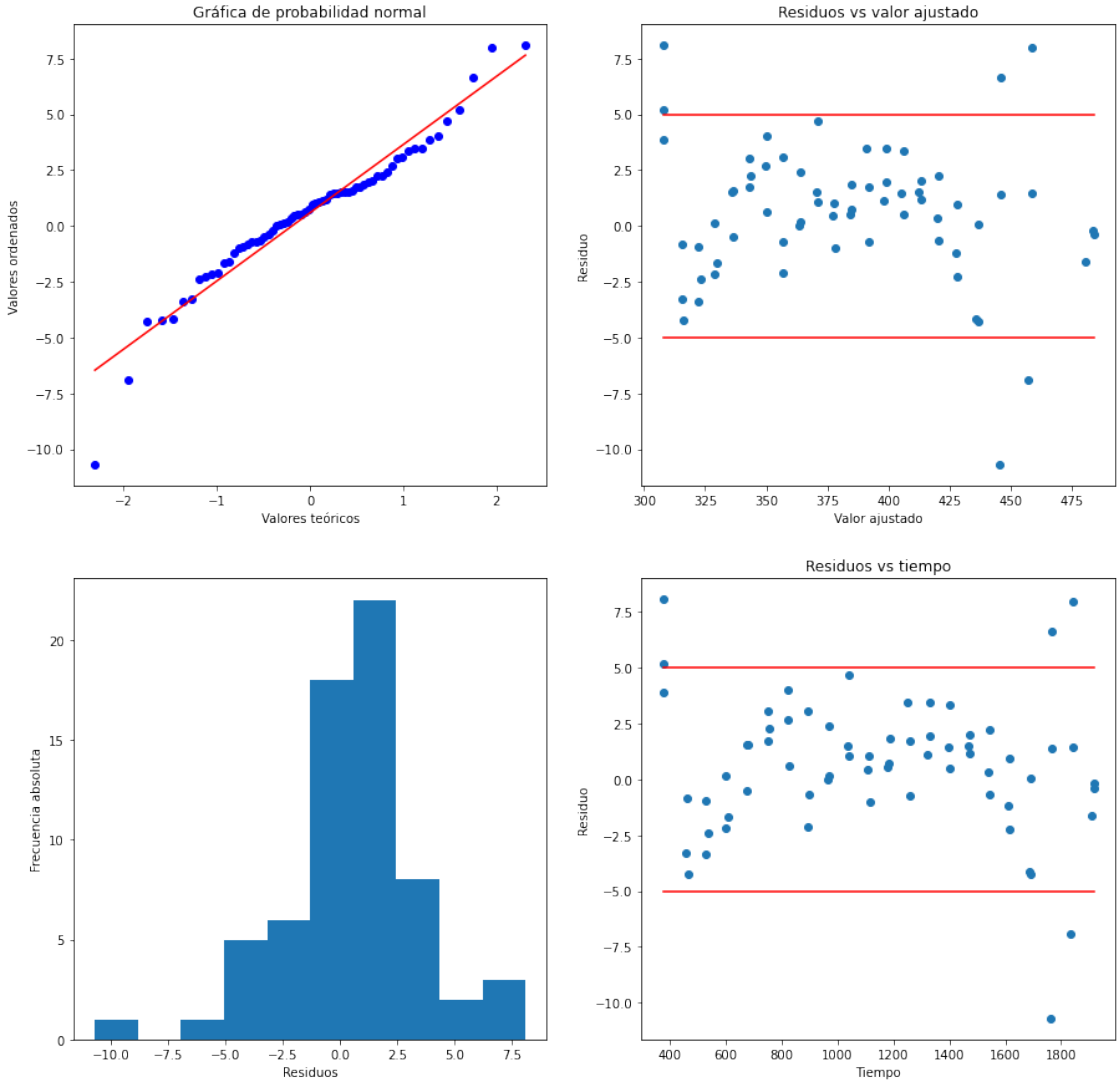


[57]:
```
anderson(residuosTeórico,dist='norm').
↪statistic,anderson(residuosTeórico,dist='norm').critical_values
```

[57]:
```
(0.8660071381102625, array([0.546, 0.622, 0.746, 0.87 , 1.035]))
```

[58]:
```
fig4,((ax1,ax2),(ax3,ax4))=plt.subplots(2,2,figsize=(15,15))
stats.probplot(residuosTeórico,dist='norm',plot=ax1)
ax1.set_title('Gráfica de probabilidad normal')
ax1.set_xlabel('Valores teóricos')
ax1.set_ylabel('Valores ordenados')
```

```
ax2.scatter(ajusT,residuosTeórico)
ax2.plot(ajusT,np.zeros_like(ajusT)+5,color='red')
ax2.plot(ajusT,np.zeros_like(ajusT)-5,color='red')
ax2.set_title('Residuos vs valor ajustado')
ax2.set_ylabel('Residuo')
ax2.set_xlabel('Valor ajustado')
ax3.hist(residuosTeórico)
ax3.set_xlabel('Residuos')
ax3.set_ylabel('Frecuencia absoluta')
ax4.scatter(corridas[:,1],residuosTeórico)
ax4.plot(corridas[:,1],np.zeros_like(corridas[:,1])+5,color='red')
ax4.plot(corridas[:,1],np.zeros_like(corridas[:,1])-5,color='red')
ax4.set_title('Residuos vs tiempo')
ax4.set_ylabel('Residuo')
ax4.set_xlabel('Tiempo')
fig4.suptitle('Diagnósticos de los residuos del modelo teórico')
plt.savefig('DiagnósticoTeóricoCorridas.png')
plt.show()
```



[59]:
```
residuosTeórico.std()**2/residuosEmpíricoPoli.std()**2
```

[59]: 1.0186119037619383

[60]:
```
1-stats.f.cdf(residuosTeórico.std()**2/residuosEmpíricoPoli.
↪std()**2,3*22-6,22-4)
```

[60]: 0.5081592379751322

5 Propuesta de depuración de datos de datos

Se propone eliminar los datos que producen residuos grandes, para determinar cuáles pueden ser residuos grandes, se observa que la mayoría de los residuos están entre -5 y 5, por lo que se propone eliminar los residuos cuyo valor absoluto sea mayor a 5, estos serían un total de 36 datos.

[61]:
```
depurando=corridas[np.where(np.abs(residuosTeórico)>5)]
depurado=corridas.copy()
```

[62]:
```
depurado=depurado.tolist()
depurando=depurando.tolist()
```

[63]:
```
len(depurando)
```

[63]:
```
6
```

[66]:
```
i=0
p=0

while i in range(len(depurado)):
    if depurado[i].tolist() in depurando:
        depurado=np.delete(depurado,i,axis=0)
        i=0
    else:
        i+=1
```

[67]:
```
len(depurado)
```

[67]:
```
60
```

[68]:
```
# coefsT,_=sopt.curve_fit(Teórico,depurado[:,1],depurado[:,2])
ajusT=Teórico(depurado[:,1],coefsT[0],coefsT[1],coefsT[2],coefsT[3],coefsT[4])
residuosTeórico=ajusT-depurado[:,2]
coefsT
```

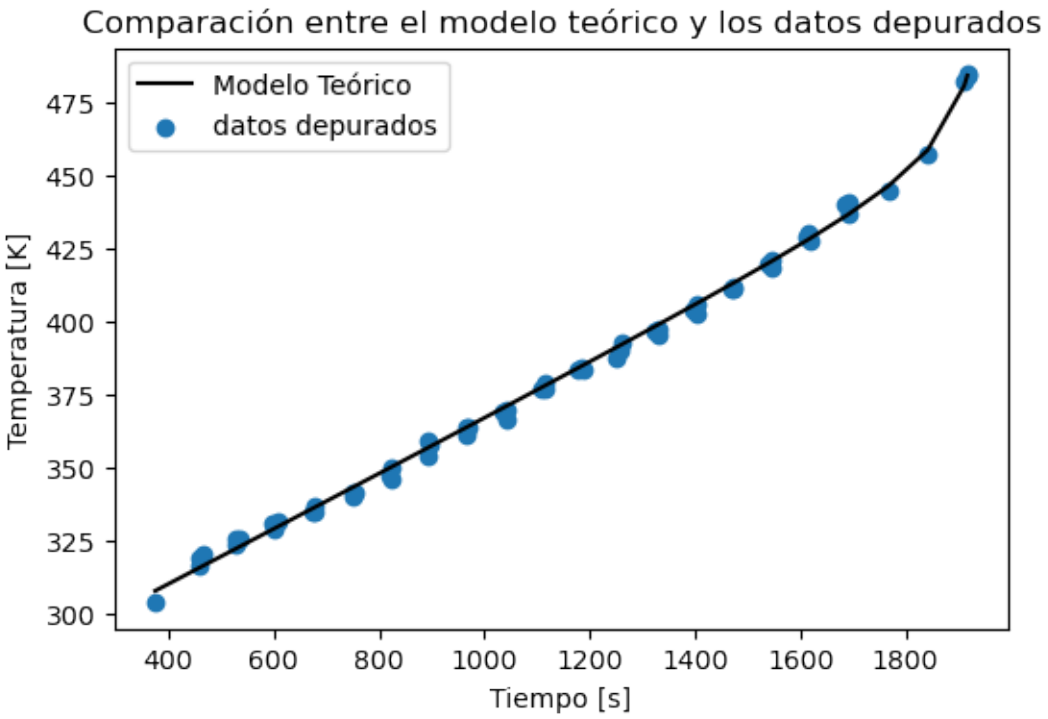
[68]:
```
array([ 1.64996382e+03, -1.77328215e+00, -1.02491714e+07, -3.75993309e+04,
       -1.89996118e+01])
```

[69]:
```
se2T2=(residuosTeórico**2).sum()/(3*len(residuosTeórico)-6)
np.sqrt(se2T2)
```

[69]:
```
1.2635377020281555
```

[70]:
```
plt.figure(dpi=100)
plt.plot(depurado[:,1],ajusT,color='black')
plt.scatter(depurado[:,1],depurado[:,2])
plt.xlabel('Tiempo [s]')
plt.ylabel('Temperatura [K]')
```

```
plt.legend(['Modelo Teórico','datos depurados'])
plt.title('Comparación entre el modelo teórico y los datos depurados')
plt.savefig('ModeloTeóricovsCorridasDepurado.png')
plt.show()
```

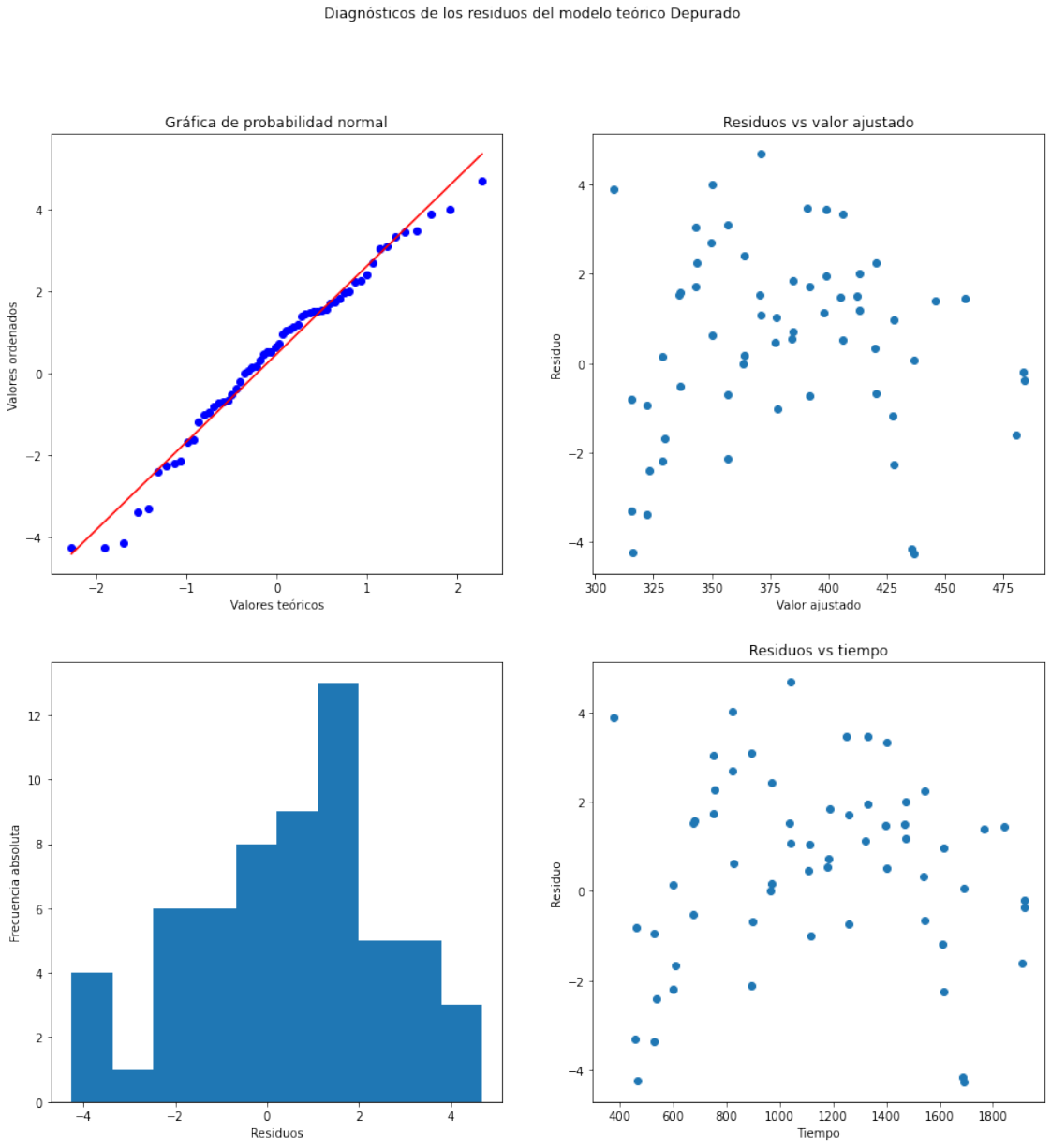


[71]:
```
anderson(residuosTeórico,dist='norm').
 ↪statistic,anderson(residuosTeórico,dist='norm').critical_values
```

[71]:
```
(0.3945677808271171, array([0.544, 0.619, 0.743, 0.866, 1.03 ]))
```

[72]:
```
fig4,((ax1,ax2),(ax3,ax4))=plt.subplots(2,2,figsize=(15,15))
stats.probplot(residuosTeórico,dist='norm',plot=ax1)
ax1.set_title('Gráfica de probabilidad normal')
ax1.set_xlabel('Valores teóricos')
ax1.set_ylabel('Valores ordenados')
ax2.scatter(ajusT,residuosTeórico)
ax2.set_title('Residuos vs valor ajustado')
ax2.set_ylabel('Residuo')
ax2.set_xlabel('Valor ajustado')
ax3.hist(residuosTeórico)
ax3.set_xlabel('Residuos')
ax3.set_ylabel('Frecuencia absoluta')
ax4.scatter(depurado[:,1],residuosTeórico)
```

```
ax4.set_title('Residuos vs tiempo')
ax4.set_ylabel('Residuo')
ax4.set_xlabel('Tiempo')
fig4.suptitle('Diagnósticos de los residuos del modelo teórico Depurado')
plt.savefig('DiagnósticoTeóricoCorridasDepurado.png')
plt.show()
```



[73]:
```
(3*len(depurado)/3-5)
```

[73]: 55.0

[74]:
```
residuosTeórico.std()
```

[74]: 2.098120385456533

[75]:
```
residuosTeórico.std()**2*(3*len(depurado)/3-5)/5**2
```

[75]: 9.684640134110197