R407 A SpecificHeat Characterization V1

March 24, 2020

[1]:
```
#Import the necessary libraries
import math as math
import numpy as np
import pandas as pd
import pickle
import itertools
from CoolProp.CoolProp import PropsSI
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
from sklearn import linear_model
from pandas import DataFrame
from mpl_toolkits.mplot3d import Axes3D
from pandas import ExcelWriter
from IPython.core.interactiveshell import InteractiveShell
import statsmodels.formula.api as sm
InteractiveShell.ast_node_interactivity = "all"
%matplotlib inline
```

[2]:
```
# Iteration parameters
fluid = 'R407A.mix'
P_min_kPa = 100 #Minimum pressure in kPa we will be evaluating
P_max_kPa = 3000 #Maximum pressure in kPa we will be evaluating
T_min_K = -30+273 #Minimum temp
T_max_K = 60+273 #Max temp
n = 2900 #Number of samples we will collect
m = 180
P_kPa = np.linspace(P_min_kPa, P_max_kPa, n) #The array of the pressures we␣
↪will be evaluating
T_K = np.linspace(T_min_K, T_max_K, m) # This is the array of temperatures␣
↪divided into m spaces
Spec = [] # create a list
combo = list(itertools.product(P_kPa,T_K)) # assign to combo the combination of␣
↪pressures and arrays
```

[3]:
```
# This step is only for extraction of data, if there is already a ' xxxx.p'␣
↪file, then it is not necesary to load data again
```

```
#Test values to make sure there was a coherent answer according to R407a␣
↪tables, since those are mostly experimental tests
# the function C fro propsSi calculates the Mass specific constant pressure␣
↪specific heat in [J/kg/K]
#print("Specific heat at 333K and 3000 kPa :", PropsSI('C', 'T', 273+60, 'P',␣
↪3000*1000, 'REFPROP::R407a.mix'), "J/kg/K") #Maxi
#print("Specific heat at 243K and 100 kPa :", PropsSI('C', 'T', 273-30, 'P',␣
↪100*1000, 'REFPROP::R407a.mix'), "J/kg/K") # Minimi



#for i in combo: # extract data from propsi, use only if library (coolprops) is␣
↪loaded and functioning
 #   [p,t]= i
  #  try:
   #     Spec.append(PropsSI('C','P', p*1000,'T',t, 'REFPROP::R407a.mix')) #␣
↪use the function PropsSI to
        # get the specific heat according to  pressure defined and temperature␣
↪defined, the pressure array has 2900 points
        #according to 3000 kPa - 100 kPa Range - the temperature similarly has␣
↪90 points
   # except:
   #     Spec.append(0) # we don't know if there is values in all the spaces,␣
↪or the extent of the ProposSI Function

        # What happened above is to make sure or assess if there is data in the␣
↪file that will adjust ot our workspace
```

[4]:
```
#print(Spec) # This is Specific heat, I have used the variable Spec because it␣
↪wass easy to start with and have a reference
```

[5]:
```
# run this only when you want to extract data again from propsi, if there is␣
↪already a SpecificHeat.p file; there is no need to run
#pickle.dump(Spec, open( "SpecificHeat.p", "wb" ) ) # save values from propsi␣
↪in file so we don't have to run each time
```

[3]:
```
X, Y = np.meshgrid(T_K,P_kPa)
```

[4]:
```
Spec = pickle.load( open( "SpecificHeat.p", "rb" ) ) # load values from file
```

[4]:
```
PropsSI('C','P',1162*1000,'T',264, 'REFPROP::R407a.mix')
```

[4]: 1358.0620567466892

[5]:
```
X.shape
Y.shape
```

```
Y
```

[5]: (2900, 180)

[5]: (2900, 180)

```
[5]: array([[  100.        ,   100.        ,   100.        , …,
          100.        ,   100.        ,   100.        ],
       [  101.00034495,   101.00034495,   101.00034495, …,
          101.00034495,   101.00034495,   101.00034495],
       [  102.00068989,   102.00068989,   102.00068989, …,
          102.00068989,   102.00068989,   102.00068989],
       …,
       [ 2997.99931011,  2997.99931011,  2997.99931011, …,
         2997.99931011,  2997.99931011,  2997.99931011],
       [ 2998.99965505,  2998.99965505,  2998.99965505, …,
         2998.99965505,  2998.99965505,  2998.99965505],
       [ 3000.        ,  3000.        ,  3000.        , …,
         3000.        ,  3000.        ,  3000.        ]])
```

[6]:
```
Z = np.split(np.array(Spec),2900)
Z = np.array(Z)
Z.shape
```

[6]: (2900, 180)

[7]:
```
# plotting of raw data to visualize the behaviour and see Cp (propsSI) outputs
from mpl_toolkits.mplot3d import Axes3D # import libraries for drawing in 3D
import matplotlib.pyplot as plt
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FormatStrFormatter
# this is a graph of the data we are dealing with
%matplotlib inline

threedee = plt.figure().gca(projection='3d')
threedee.scatter(X,Y, Z)
threedee.set_xlabel('Pressure (kPa)')
threedee.set_ylabel('Temp. (K)')
threedee.set_zlabel('Cp(J/kg/K)')
plt.show()
#fig = plt.figure()
#ax = fig.gca(projection='3d')
#surf = ax.plot_surface(X, Y, Z, cmap=cm.coolwarm, linewidth=0,␣
 ↪antialiased=False)# plot the plot will give two distinc surfaces with an␣
 ↪area in between
#ax.set_xlabel('T (K)', fontsize = 14)
#ax.set_ylabel('P(kPa)', fontsize = 14)
```

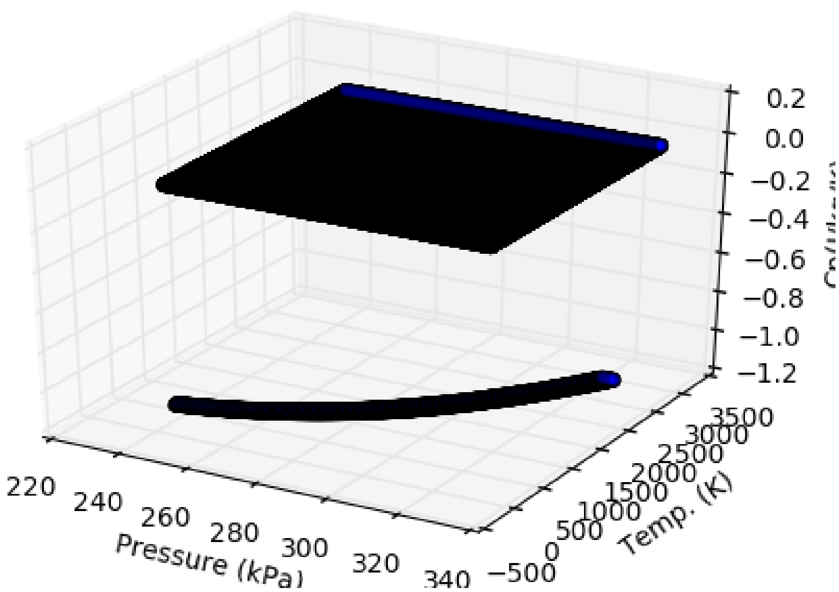
```
#ax.set_zlabel('C (kJ/kg/K)', fontsize = 14)
```

[7]:
```
<mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x7fa8af4bb518>
```

[7]:
```
<matplotlib.text.Text at 0x7fa8af4f8860>
```

[7]:
```
<matplotlib.text.Text at 0x7fa8af50c390>
```

[7]:
```
<matplotlib.text.Text at 0x7fa8af512d68>
```



[8]:
```
shf= pd.DataFrame.from_records( combo, columns= ['Pressure','Temperature'])
shf = shf.assign(Cp = Spec)
len (shf)
```

[8]: 522000

[9]:
```
shf.head(3)
```

[9]:

	Pressure	Temperature	Cp
0	100.0	243.000000	770.671210
1	100.0	243.502793	770.801452
2	100.0	244.005587	770.950061

[10]:
```
Mask = (shf>0)
```

[11]:
```
shf[Mask]
```

[11]:

	Pressure	Temperature	Cp
0	100.0	243.000000	770.671210
1	100.0	243.502793	770.801452
2	100.0	244.005587	770.950061
3	100.0	244.508380	771.116277
4	100.0	245.011173	771.299379
5	100.0	245.513966	771.498686
6	100.0	246.016760	771.713552
7	100.0	246.519553	771.943363
8	100.0	247.022346	772.187539
9	100.0	247.525140	772.445528
10	100.0	248.027933	772.716807
11	100.0	248.530726	773.000880
12	100.0	249.033520	773.297272
13	100.0	249.536313	773.605537
14	100.0	250.039106	773.925246
15	100.0	250.541899	774.255994
16	100.0	251.044693	774.597393
17	100.0	251.547486	774.949075
18	100.0	252.050279	775.310689
19	100.0	252.553073	775.681900
20	100.0	253.055866	776.062390
21	100.0	253.558659	776.451853
22	100.0	254.061453	776.849999
23	100.0	254.564246	777.256549
24	100.0	255.067039	777.671239
25	100.0	255.569832	778.093814
26	100.0	256.072626	778.524031
27	100.0	256.575419	778.961659
28	100.0	257.078212	779.406474
29	100.0	257.581006	779.858264
…	…	…	…
521970	3000.0	318.418994	NaN
521971	3000.0	318.921788	NaN
521972	3000.0	319.424581	NaN
521973	3000.0	319.927374	NaN
521974	3000.0	320.430168	NaN
521975	3000.0	320.932961	NaN
521976	3000.0	321.435754	NaN
521977	3000.0	321.938547	NaN
521978	3000.0	322.441341	NaN
521979	3000.0	322.944134	NaN
521980	3000.0	323.446927	NaN
521981	3000.0	323.949721	NaN
521982	3000.0	324.452514	NaN
521983	3000.0	324.955307	NaN
521984	3000.0	325.458101	NaN

```
521985    3000.0   325.960894         NaN
521986    3000.0   326.463687         NaN
521987    3000.0   326.966480         NaN
521988    3000.0   327.469274         NaN
521989    3000.0   327.972067         NaN
521990    3000.0   328.474860         NaN
521991    3000.0   328.977654         NaN
521992    3000.0   329.480447         NaN
521993    3000.0   329.983240         NaN
521994    3000.0   330.486034         NaN
521995    3000.0   330.988827         NaN
521996    3000.0   331.491620         NaN
521997    3000.0   331.994413         NaN
521998    3000.0   332.497207         NaN
521999    3000.0   333.000000         NaN

[522000 rows x 3 columns]
```

[12]:
```
groupingtest = shf.loc[shf['Cp']>0]
#len(shf.loc[shf['Cp']>0])
groupingtest.head(2)
```

[12]:

	Pressure	Temperature	Cp
0	100.0	243.000000	770.671210
1	100.0	243.502793	770.801452

[ ]:
```
#writer = pd.ExcelWriter('pandas_simplecp.xlsx', engine='xlsxwriter')
#shf.loc[shf['Cp']<0].to_excel(writer, sheet_name='Sheet1')
#writer.save()
```

[13]:
```
negatives = shf.loc[shf['Cp']<0]
positives = shf.loc[shf['Cp']>0]
```

[14]:
```
# Plot the extracted data from propsi
print(shf.head());
threedee = plt.figure().gca(projection='3d')
threedee.scatter(positives['Temperature'],positives['Pressure'],␣
 ↪positives['Cp'])
threedee.set_xlabel('Temp. (K)')
threedee.set_ylabel('Pressure (kPa)')
threedee.set_zlabel('Cp (J/kg/k)')
plt.show()
```

```
   Pressure  Temperature          Cp
0     100.0   243.000000  770.671210
1     100.0   243.502793  770.801452
2     100.0   244.005587  770.950061
```

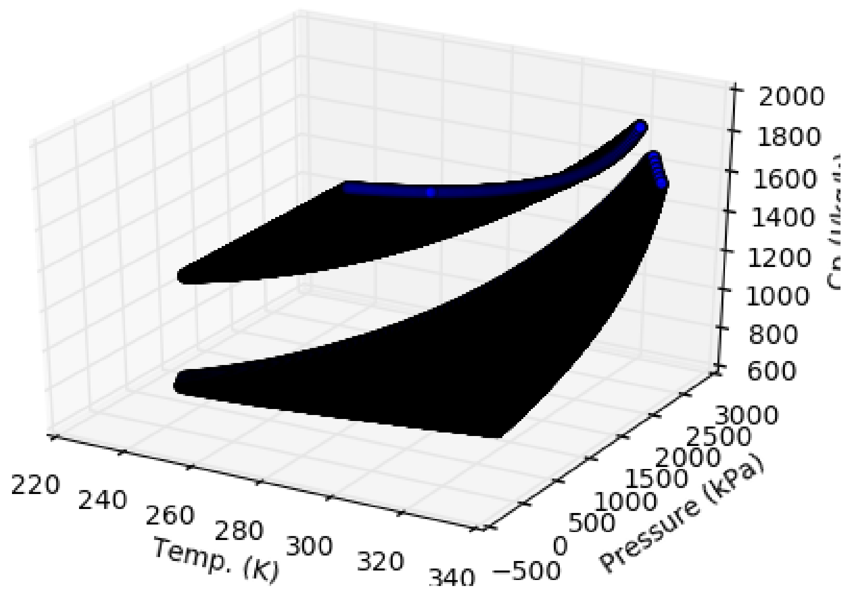
```
3      100.0    244.508380   771.116277
4      100.0    245.011173   771.299379
```

[14]: <mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x7fa8a542aac8>

[14]: <matplotlib.text.Text at 0x7fa8a5bb22e8>

[14]: <matplotlib.text.Text at 0x7fa8a5bb7dd8>

[14]: <matplotlib.text.Text at 0x7fa8a5bc87f0>



[ ]:
```
# Time to separate the characterization!! :)
```

Regression model for Mix (acording to saturated caracterization)

0.0.1 Characterization model for latent heat portion liquid portion

[15]:
```
negatives.head(4) # negatives are the values selected to be outliers from the␣
 ↪'normal' Cp value
Latent_negatives = negatives[(negatives['Temperature']> 148.074237046 + 25.
 ↪495200*negatives['Pressure']**0.24854063)&(negatives['Temperature'] < 151.12␣
 ↪+ 27.9058*negatives['Pressure']**0.237508)]
threedee = plt.figure().gca(projection='3d')
threedee.
 ↪scatter(Latent_negatives['Temperature'],Latent_negatives['Pressure'],Latent_negatives['Cp']
threedee.set_xlabel('Temperature (K)')
```

```
threedee.set_ylabel('Pressure (kPa)')
threedee.set_zlabel('Cp (J/kg/K)')
plt.show()
```

[15]:

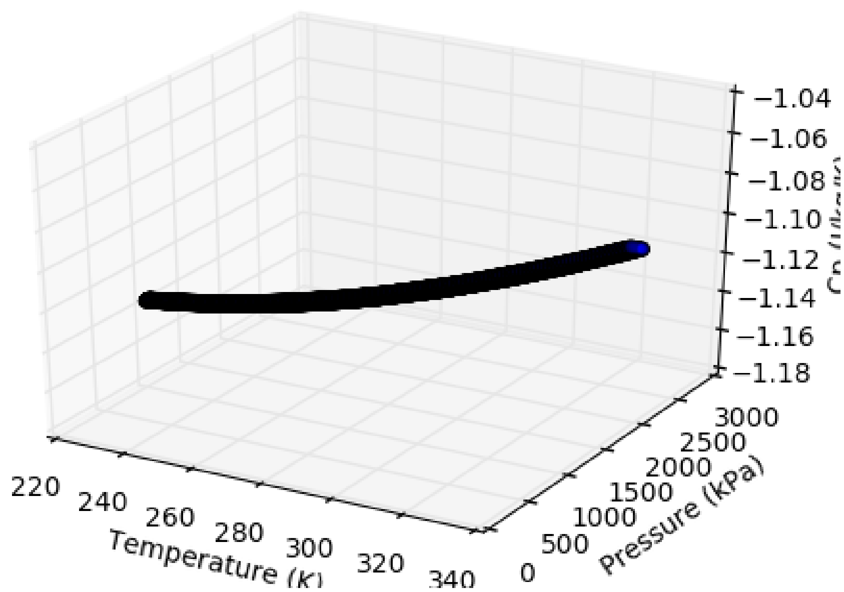
	Pressure	Temperature	Cp
9360	152.017937	243.0	-1.109744e+08
9540	153.018282	243.0	-1.109744e+08
9720	154.018627	243.0	-1.109744e+08
9900	155.018972	243.0	-1.109744e+08

[15]: <mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x7fa8a41e6dd8>

[15]: <matplotlib.text.Text at 0x7fa8a5bdd2e8>

[15]: <matplotlib.text.Text at 0x7fa8a65ccba8>

[15]: <matplotlib.text.Text at 0x7fa8a41b6470>



[ ]:
```
# Due to the results, it is clear that propsSI for the latent portion of the
 ↪specific heat data, is throwing a value that does not match with the
 ↪physical phenomena.
# The value for the portion should be instead in [J/mol] or [J/kg] since it is
 ↪a rise described as the mass or molar latent heat at constant temperature
# Therefore, this portion of the characterization will be performed in a
 ↪seperate krunchy book
```

0.1 Characterization model Specific Heat for liquid portion

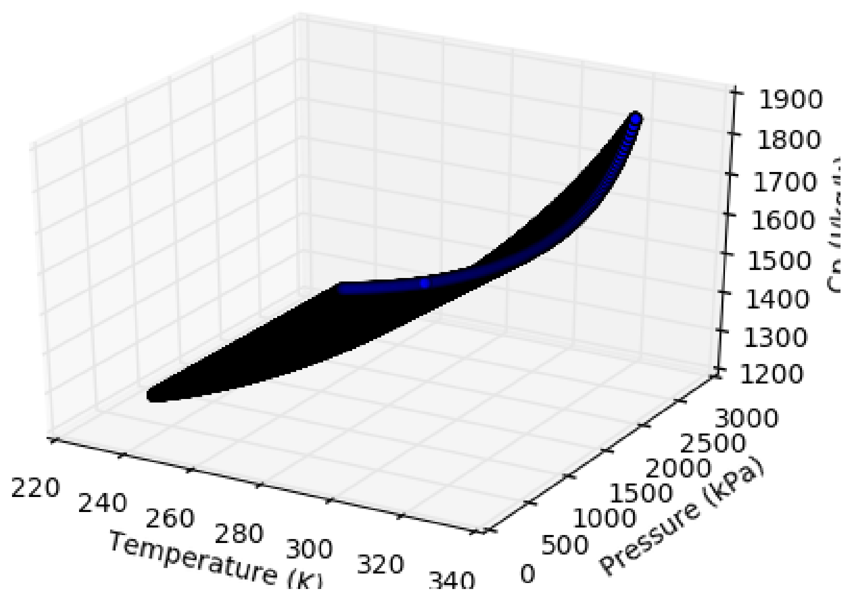
[16]:
```
Liquid_positives = positives[(positives['Temperature'] < 148.074237046 + 25.
→495200534*positives['Pressure']**0.248540630499)]
threedee = plt.figure().gca(projection='3d')
threedee.
→scatter(Liquid_positives['Temperature'],Liquid_positives['Pressure'],Liquid_positives['Cp']
threedee.set_xlabel('Temperature (K)')
threedee.set_ylabel('Pressure (kPa)')
threedee.set_zlabel('Cp (J/kg/k)')
plt.show()
```

[16]: <mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x7fa8ab4450f0>

[16]: <matplotlib.text.Text at 0x7fa8a41c2780>

[16]: <matplotlib.text.Text at 0x7fa8ab3ab940>

[16]: <matplotlib.text.Text at 0x7fa8ab3ba358>



[17]:
```
Liquid_positives = Liquid_positives.assign(Temp_sq =␣
→Liquid_positives['Temperature']**2)#1.56
Liquid_positives = Liquid_positives.assign(Press_sq =␣
→Liquid_positives['Pressure']**2) #1.4
Liquid_positives.head(3)
```

[17]:

```
            Pressure  Temperature           Cp  Temp_sq      Press_sq
17820  199.034150        243.0  1310.394142  59049.0  39614.592750
18000  200.034495        243.0  1310.390232  59049.0  40013.799051
18180  201.034840        243.0  1310.386321  59049.0  40415.006733
```

[39]:

```python
resultLiquid = sm.ols(formula="Cp ~ Temperature + Pressure + Temp_sq",␣
 ↪data=Liquid_positives).fit()
forecastLiquid = resultLiquid.
 ↪predict(Liquid_positives[['Temperature','Pressure','Temp_sq']])
ErrorLiquid = pd.DataFrame({'Error': forecastLiquid-Liquid_positives.Cp})

print(resultLiquid.params)
print(resultLiquid.summary())
ErrorLiquid.describe()
```

```
Intercept      5006.961681
Temperature     -30.457731
Pressure         -0.008074
Temp_sq           0.063061
dtype: float64
                            OLS Regression Results
==============================================================================
Dep. Variable:                     Cp   R-squared:                       0.991
Model:                            OLS   Adj. R-squared:                  0.991
Method:                 Least Squares   F-statistic:                 9.083e+06
Date:                Wed, 13 Dec 2017   Prob (F-statistic):               0.00
Time:                        19:12:34   Log-Likelihood:            -9.0782e+05
No. Observations:              249026   AIC:                         1.816e+06
Df Residuals:                  249022   BIC:                         1.816e+06
Df Model:                           3
Covariance Type:            nonrobust
===============================================================================
                  coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------
Intercept    5006.9617      3.377   1482.485      0.000    5000.342    5013.581
Temperature   -30.4577      0.024  -1249.461      0.000     -30.506     -30.410
Pressure       -0.0081   3.47e-05   -232.643      0.000      -0.008      -0.008
Temp_sq         0.0631   4.39e-05   1437.625      0.000       0.063       0.063
==============================================================================
Omnibus:                   133221.287   Durbin-Watson:                   0.155
Prob(Omnibus):                  0.000   Jarque-Bera (JB):          2791171.881
Skew:                           2.127   Prob(JB):                         0.00
Kurtosis:                      18.840   Cond. No.                     1.39e+07
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
```

specified.
[2] The condition number is large, 1.39e+07. This might indicate that there are strong multicollinearity or other numerical problems.

[39]:

```
              Error
count  2.490260e+05
mean  -4.790373e-10
std    9.267901e+00
min   -1.045214e+02
25%   -5.547611e+00
50%   -1.018039e+00
75%    5.538027e+00
max    2.151780e+01
```

[40]:

```python
plt.hist(ErrorLiquid.Error, bins = 1500); # histogram of error distribution
plt.title('Error distribution')
plt.xlabel('Spread from function Cp (J/kg/K)))')
plt.ylabel('Frecuency')
```

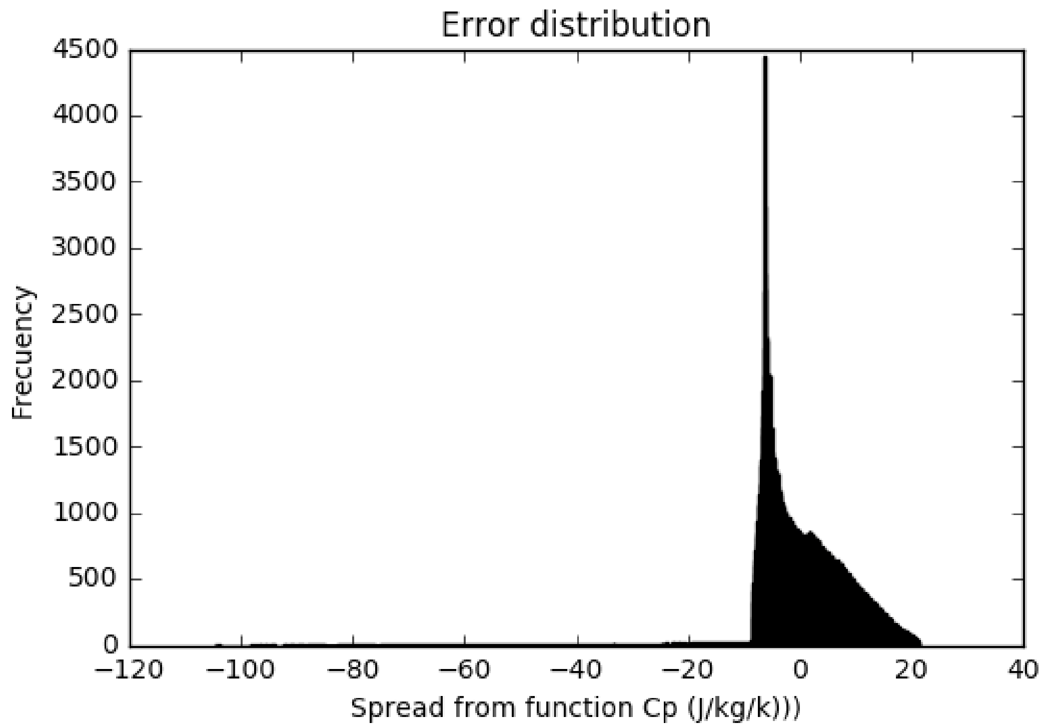
[40]:

```
(array([  1.,   1.,   0., …,  29.,  23.,  14.]),
 array([-104.52140613, -104.43737999, -104.35335385, …,   21.34974705,
          21.43377318,   21.51779932]),
 <a list of 1500 Patch objects>)
```

[40]: `<matplotlib.text.Text at 0x7fa892b5efd0>`

[40]: `<matplotlib.text.Text at 0x7fa8930e70b8>`

[40]: `<matplotlib.text.Text at 0x7fa892e48eb8>`



1 Result for liquid portion CpLiquid = aT^2+bT+cP+f [J/kg/K]

1.0.1 R2 = 0.991

a is the parameter for Temp_sq : 0.063061

b is the parameter for Temperature : -30.457731

c is the parameter for Pressure : -0.008074

f is the Intercept: 5006.961681

2 Result for liquid portion CpLiquid = bT+cP+f

2.0.1 R2 = 0.916

b is the parameter for Temperature : 4.558210

c is the parameter for Pressure : -0.002865

f is the Intercept: 165.200746

3 Result for liquid portion CpLiquid = bT+f

3.0.1 R2 = 0.916

b is the parameter for Temperature : 4.525411

f is the Intercept: 169.546153

3.1 Characterization model for Specific Heat for Vapor portion

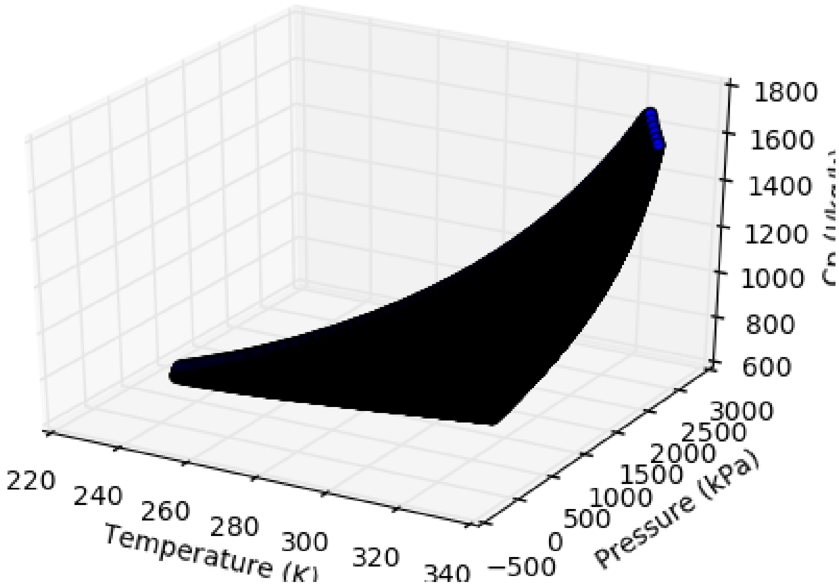
[24]:
```
Vapor_positives = positives[(positives['Temperature'] > 151.12884541 + 27.
↪9058175606*positives['Pressure']**0.23750882839)]
threedee = plt.figure().gca(projection='3d')
threedee.
↪scatter(Vapor_positives['Temperature'],Vapor_positives['Pressure'],Vapor_positives['Cp'])
threedee.set_xlabel('Temperature (K)')
threedee.set_ylabel('Pressure (kPa)')
threedee.set_zlabel('Cp (J/kg/k)')
plt.show()
```

[24]: <mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x7fa8a9d74ac8>

[24]: <matplotlib.text.Text at 0x7fa8aa20a518>

[24]: <matplotlib.text.Text at 0x7fa8aa0aaef0>

[24]: <matplotlib.text.Text at 0x7fa8aa033358>



[25]:
```
Vapor_positives.head(3)
```

[25]:

	Pressure	Temperature	Cp
0	100.0	243.000000	770.671210
1	100.0	243.502793	770.801452
2	100.0	244.005587	770.950061

[34]:
```
Vapor_positives = Vapor_positives.assign(Temp_sq =␣
↪Vapor_positives['Temperature']**2)#1.56 as a suggestion, overall it is␣
↪reasonable
Vapor_positives = Vapor_positives.assign(Press_sq =␣
↪Vapor_positives['Pressure']**2) #1.4
```

[35]:
```
resultVapor = sm.ols(formula="Cp ~ Temperature + Pressure ",␣
↪data=Vapor_positives).fit()
forecastVapor = resultVapor.predict(Vapor_positives[['Pressure','Temperature']])
ErrorVapor = pd.DataFrame({'Error': forecastVapor-Vapor_positives.Cp})

print(resultVapor.params)
print(resultVapor.summary())
ErrorVapor.describe()
```

```
Intercept      887.750263
Temperature     -0.381473
Pressure         0.276162
dtype: float64
                            OLS Regression Results
```

```
==============================================================================
Dep. Variable:                     Cp   R-squared:                       0.937
Model:                            OLS   Adj. R-squared:                  0.937
Method:                 Least Squares   F-statistic:                 1.206e+06
Date:                Wed, 13 Dec 2017   Prob (F-statistic):               0.00
Time:                        19:10:22   Log-Likelihood:            -8.3675e+05
No. Observations:              163429   AIC:                         1.673e+06
Df Residuals:                  163426   BIC:                         1.674e+06
Df Model:                           2
Covariance Type:            nonrobust
===============================================================================
                  coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------
Intercept     887.7503      1.788    496.464      0.000     884.246     891.255
Temperature    -0.3815      0.006    -62.475      0.000      -0.393      -0.370
Pressure        0.2762      0.000   1307.812      0.000       0.276       0.277
==============================================================================
Omnibus:                    41533.631   Durbin-Watson:                   0.177
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           169829.334
Skew:                           1.209   Prob(JB):                         0.00
Kurtosis:                       7.369   Cond. No.                     1.88e+04
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.88e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

[35]:

	Error
count	1.634290e+05
mean	4.501923e-13
std	4.048774e+01
min	-2.775680e+02
25%	-1.836774e+01
50%	3.330836e+00
75%	2.282027e+01
max	8.744900e+01

[33]:
```
plt.hist(ErrorVapor.Error, bins = 150); # histogram of error distribution
plt.title('Error distribution')
plt.xlabel('Spread from function Cp (J/kg/k)))')
plt.ylabel('Frecuency')
```

[33]:
```
(array([  3.00000000e+00,   4.00000000e+00,   4.00000000e+00,
          4.00000000e+00,   5.00000000e+00,   6.00000000e+00,
          6.00000000e+00,   6.00000000e+00,   6.00000000e+00,
```

```
7.00000000e+00,   8.00000000e+00,   1.00000000e+01,
1.00000000e+01,   1.20000000e+01,   1.40000000e+01,
1.20000000e+01,   1.20000000e+01,   1.40000000e+01,
1.50000000e+01,   1.40000000e+01,   1.60000000e+01,
2.10000000e+01,   2.00000000e+01,   2.00000000e+01,
2.00000000e+01,   2.30000000e+01,   2.30000000e+01,
2.30000000e+01,   2.60000000e+01,   2.60000000e+01,
2.60000000e+01,   3.20000000e+01,   3.10000000e+01,
3.10000000e+01,   3.70000000e+01,   3.40000000e+01,
3.60000000e+01,   3.60000000e+01,   3.90000000e+01,
4.00000000e+01,   4.50000000e+01,   4.50000000e+01,
4.60000000e+01,   5.00000000e+01,   4.90000000e+01,
5.00000000e+01,   5.60000000e+01,   5.50000000e+01,
5.90000000e+01,   6.30000000e+01,   6.20000000e+01,
6.70000000e+01,   6.80000000e+01,   7.00000000e+01,
7.30000000e+01,   7.60000000e+01,   7.70000000e+01,
7.70000000e+01,   8.40000000e+01,   8.90000000e+01,
8.80000000e+01,   9.30000000e+01,   9.50000000e+01,
1.02000000e+02,   9.90000000e+01,   1.06000000e+02,
1.16000000e+02,   1.10000000e+02,   1.23000000e+02,
1.22000000e+02,   1.31000000e+02,   1.34000000e+02,
1.40000000e+02,   1.43000000e+02,   1.56000000e+02,
1.62000000e+02,   1.68000000e+02,   2.08000000e+02,
2.61000000e+02,   2.96000000e+02,   3.48000000e+02,
3.99000000e+02,   4.39000000e+02,   5.02000000e+02,
5.53000000e+02,   6.13000000e+02,   6.66000000e+02,
7.37000000e+02,   7.95000000e+02,   8.78000000e+02,
9.48000000e+02,   1.02900000e+03,   1.13500000e+03,
1.22300000e+03,   1.33800000e+03,   1.43500000e+03,
1.57000000e+03,   1.68500000e+03,   1.79900000e+03,
1.92700000e+03,   2.05500000e+03,   2.18200000e+03,
2.31700000e+03,   2.48400000e+03,   2.62400000e+03,
2.79200000e+03,   2.99700000e+03,   3.20100000e+03,
3.42100000e+03,   3.68800000e+03,   3.99800000e+03,
4.36800000e+03,   4.85600000e+03,   5.51500000e+03,
6.67600000e+03,   9.58600000e+03,   7.01800000e+03,
5.66400000e+03,   5.01300000e+03,   4.43500000e+03,
4.08400000e+03,   3.77500000e+03,   3.50600000e+03,
3.21200000e+03,   3.08600000e+03,   2.84000000e+03,
2.68900000e+03,   2.54500000e+03,   2.38600000e+03,
2.23000000e+03,   2.08100000e+03,   1.96000000e+03,
1.84200000e+03,   1.73800000e+03,   1.55900000e+03,
1.46500000e+03,   1.42300000e+03,   1.27300000e+03,
1.28000000e+03,   1.14400000e+03,   1.15100000e+03,
9.96000000e+02,   9.83000000e+02,   8.91000000e+02,
7.89000000e+02,   7.66000000e+02,   6.44000000e+02,
5.28000000e+02,   4.38000000e+02,   3.70000000e+02]),
```

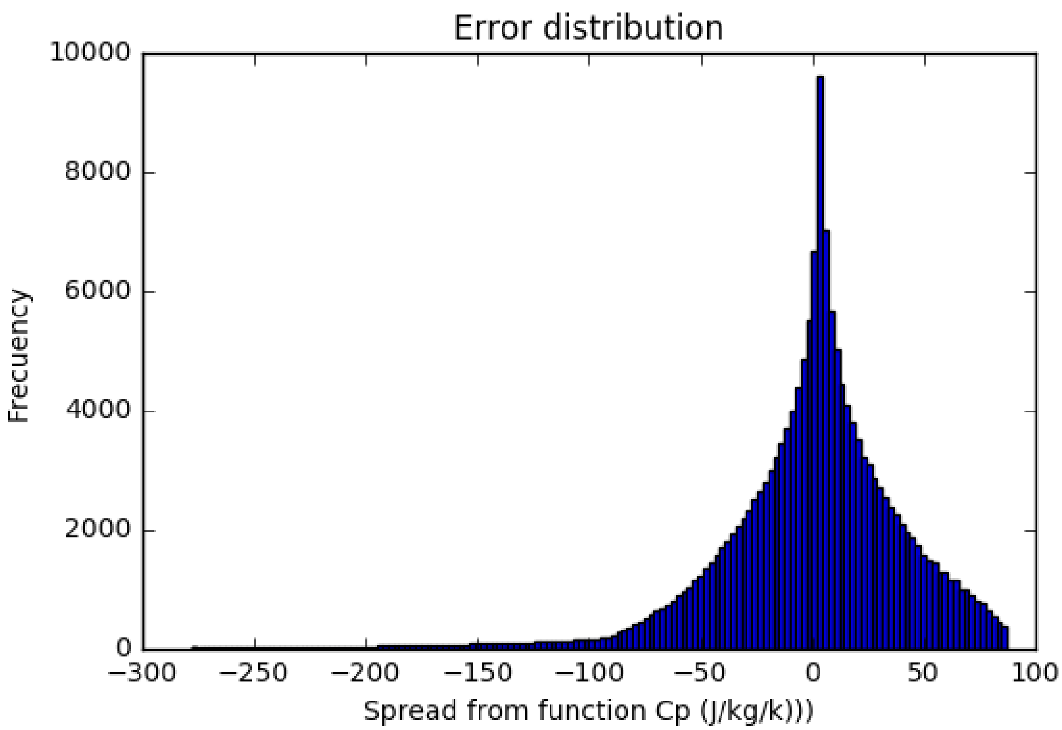
```
array([ -2.77568022e+02,  -2.75134575e+02,  -2.72701128e+02,
        -2.70267681e+02,  -2.67834234e+02,  -2.65400788e+02,
        -2.62967341e+02,  -2.60533894e+02,  -2.58100447e+02,
        -2.55667000e+02,  -2.53233553e+02,  -2.50800107e+02,
        -2.48366660e+02,  -2.45933213e+02,  -2.43499766e+02,
        -2.41066319e+02,  -2.38632872e+02,  -2.36199426e+02,
        -2.33765979e+02,  -2.31332532e+02,  -2.28899085e+02,
        -2.26465638e+02,  -2.24032191e+02,  -2.21598745e+02,
        -2.19165298e+02,  -2.16731851e+02,  -2.14298404e+02,
        -2.11864957e+02,  -2.09431510e+02,  -2.06998064e+02,
        -2.04564617e+02,  -2.02131170e+02,  -1.99697723e+02,
        -1.97264276e+02,  -1.94830829e+02,  -1.92397383e+02,
        -1.89963936e+02,  -1.87530489e+02,  -1.85097042e+02,
        -1.82663595e+02,  -1.80230148e+02,  -1.77796702e+02,
        -1.75363255e+02,  -1.72929808e+02,  -1.70496361e+02,
        -1.68062914e+02,  -1.65629467e+02,  -1.63196021e+02,
        -1.60762574e+02,  -1.58329127e+02,  -1.55895680e+02,
        -1.53462233e+02,  -1.51028786e+02,  -1.48595339e+02,
        -1.46161893e+02,  -1.43728446e+02,  -1.41294999e+02,
        -1.38861552e+02,  -1.36428105e+02,  -1.33994658e+02,
        -1.31561212e+02,  -1.29127765e+02,  -1.26694318e+02,
        -1.24260871e+02,  -1.21827424e+02,  -1.19393977e+02,
        -1.16960531e+02,  -1.14527084e+02,  -1.12093637e+02,
        -1.09660190e+02,  -1.07226743e+02,  -1.04793296e+02,
        -1.02359850e+02,  -9.99264028e+01,  -9.74929559e+01,
        -9.50595091e+01,  -9.26260623e+01,  -9.01926154e+01,
        -8.77591686e+01,  -8.53257217e+01,  -8.28922749e+01,
        -8.04588281e+01,  -7.80253812e+01,  -7.55919344e+01,
        -7.31584876e+01,  -7.07250407e+01,  -6.82915939e+01,
        -6.58581471e+01,  -6.34247002e+01,  -6.09912534e+01,
        -5.85578065e+01,  -5.61243597e+01,  -5.36909129e+01,
        -5.12574660e+01,  -4.88240192e+01,  -4.63905724e+01,
        -4.39571255e+01,  -4.15236787e+01,  -3.90902319e+01,
        -3.66567850e+01,  -3.42233382e+01,  -3.17898913e+01,
        -2.93564445e+01,  -2.69229977e+01,  -2.44895508e+01,
        -2.20561040e+01,  -1.96226572e+01,  -1.71892103e+01,
        -1.47557635e+01,  -1.23223167e+01,  -9.88886981e+00,
        -7.45542298e+00,  -5.02197614e+00,  -2.58852931e+00,
        -1.55082469e-01,   2.27836437e+00,   4.71181120e+00,
         7.14525804e+00,   9.57870488e+00,   1.20121517e+01,
         1.44455985e+01,   1.68790454e+01,   1.93124922e+01,
         2.17459391e+01,   2.41793859e+01,   2.66128327e+01,
         2.90462796e+01,   3.14797264e+01,   3.39131732e+01,
         3.63466201e+01,   3.87800669e+01,   4.12135138e+01,
         4.36469606e+01,   4.60804074e+01,   4.85138543e+01,
         5.09473011e+01,   5.33807479e+01,   5.58141948e+01,
         5.82476416e+01,   6.06810884e+01,   6.31145353e+01,
```

```
            6.55479821e+01,   6.79814290e+01,   7.04148758e+01,
            7.28483226e+01,   7.52817695e+01,   7.77152163e+01,
            8.01486631e+01,   8.25821100e+01,   8.50155568e+01,
            8.74490036e+01]),
 <a list of 150 Patch objects>)
```

```
[33]: <matplotlib.text.Text at 0x7fa8a4d44be0>

[33]: <matplotlib.text.Text at 0x7fa8a9c37630>

[33]: <matplotlib.text.Text at 0x7fa892c097f0>
```



4 Result for Vapor portion CpVapor = aT^2+bT+cP+f

4.1 [J/kg/K]

4.1.1 R2 = 0.939

a is the parameter for Temp_sq : -0.017369

b is the parameter for Temperature : 9.963247

c is the parameter for Pressure : 0.278669

f is the Intercept: -645.880275

5 Result for Vapor portion CpVapor = bT+cP+f

5.0.1 R2 = 0.937

b is the parameter for Temperature : -0.381473

c is the parameter for Pressure : 0.276162

f is the Intercept: 887.750263

[ ]:
```
#Spec.append(PropsSI('C','P', p*1000,'T',t, 'REFPROP::R407a.mix')) # test
#in case there is a need to export data to excel
#writer = pd.ExcelWriter('pandas_simpledensity.xlsx', engine='xlsxwriter')
#Liquid_df.to_excel(writer, sheet_name='Sheet1')
#writer.save()
```