Eurofer_physical

July 29, 2024

[1]:
```
import import_ipynb
import pandas as pd
import numpy as np
from IPython.display import display, Markdown, Math, HTML
from Utilities import fit_poly, display_fitting, polynomial_model, custom_plot

#-1 Load the data from the Excel file
col_names = ["Temp (K)", "Thermal Diffusivity Coef. (cm2/s)"]
df =pd.read_excel('Eurofer_physical_data.
 ↪xlsx',sheet_name="Eurofer_diffusivity", \
                  header=2, usecols=[0,1],names=col_names, nrows=10)
T, Property= [df[col].dropna().to_numpy() for col in df]

#-2 Call the function and get the output and optimized parameters
fit_output, popt = fit_poly(T, Property,2)

#-3 Display fitting parameters and LaTeX equation for Eurofer 97
display_fitting('Eurofer 97', r'\alpha', fit_output, popt)

#-4 Generate fitted data for Eurofer 97
T_fit = np.linspace(0, 1000, 200)
Prop_fit = polynomial_model(T_fit, *popt)

#-5 Plot the data
custom_plot(T, Property, T_fit, Prop_fit,
            x_label='Temperature [K]', y_label='Diffusivity [$cm^2/s$]',
            title=' Eurofer Diffusivity vs Temperature',
            data_label='Eurofer Data', fit_label='Eurofer Fit',
            scale='linear', xlim=(0, 1000), ylim=(0, 0.1))
```

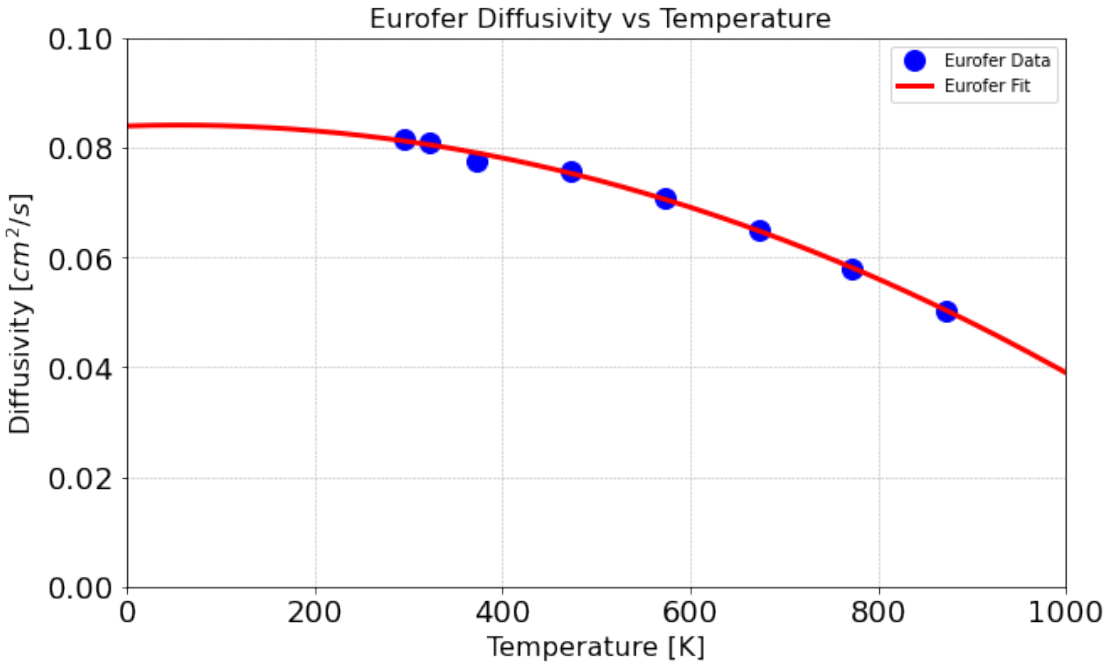
```
importing Jupyter notebook from Utilities.ipynb
```

```
<IPython.core.display.HTML object>
```

Fitting parameters for Eurofer 97

```
# Goodness-of-fit parameters

Optimized parameters: [ 8.38897309e-02  5.72789763e-06 -5.06371060e-08]
```

```
R-squared: 0.997015628990503

Reduced chi-squared: 5.38965583440041e-07

The equation for Eurofer 97 \alpha is:

<IPython.core.display.HTML object>
```



[ ]:

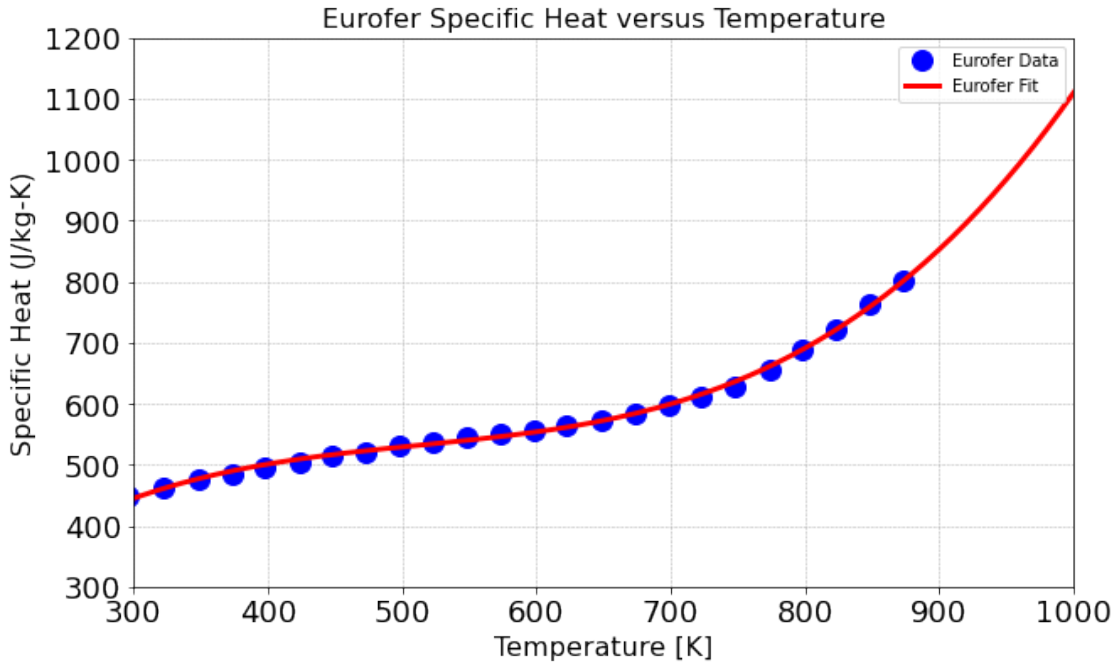
[2]:
```
#-1 Load the data from the Excel file
col_names = ["Temp (K)", "Specific Heat (J/kg-K)"]
df =pd.read_excel('Eurofer_physical_data.
  ↪xlsx',sheet_name="Eurofer_specific_heat", header=2, usecols=[0,1], nrows=28)
T,Property= [df[col].dropna().to_numpy() for col in df]

#-2 Call the function and get the output and optimized parameters
fit_output, popt = fit_poly(T, Property,3)

#-3 Display fitting parameters and LaTeX equation for Eurofer 97
display_fitting('Eurofer 97', 'C_p', fit_output, popt)
```

```
#-4 Generate fitted data for Eurofer 97
T_fit = np.linspace(0, 1000, 200)
Property_fit = polynomial_model(T_fit, *popt)

#-5 Plot the data
custom_plot(T, Property, T_fit, Property_fit,
            x_label='Temperature [K]', y_label='Specific Heat (J/kg-K)',
            title='Eurofer Specific Heat versus Temperature',
            data_label='Eurofer Data', fit_label='Eurofer Fit',
            scale='linear', xlim=(300, 1000), ylim=(300, 1200))
```

```
<IPython.core.display.HTML object>
```

Fitting parameters for Eurofer 97

```
# Goodness-of-fit parameters

Optimized parameters: [-1.38579250e+02  3.47363127e+00 -6.32751471e-03
4.10244358e-06]

R-squared: 0.9983133359160593

Reduced chi-squared: 17.04805129084386

The equation for Eurofer 97 C_p is:


<IPython.core.display.HTML object>
```



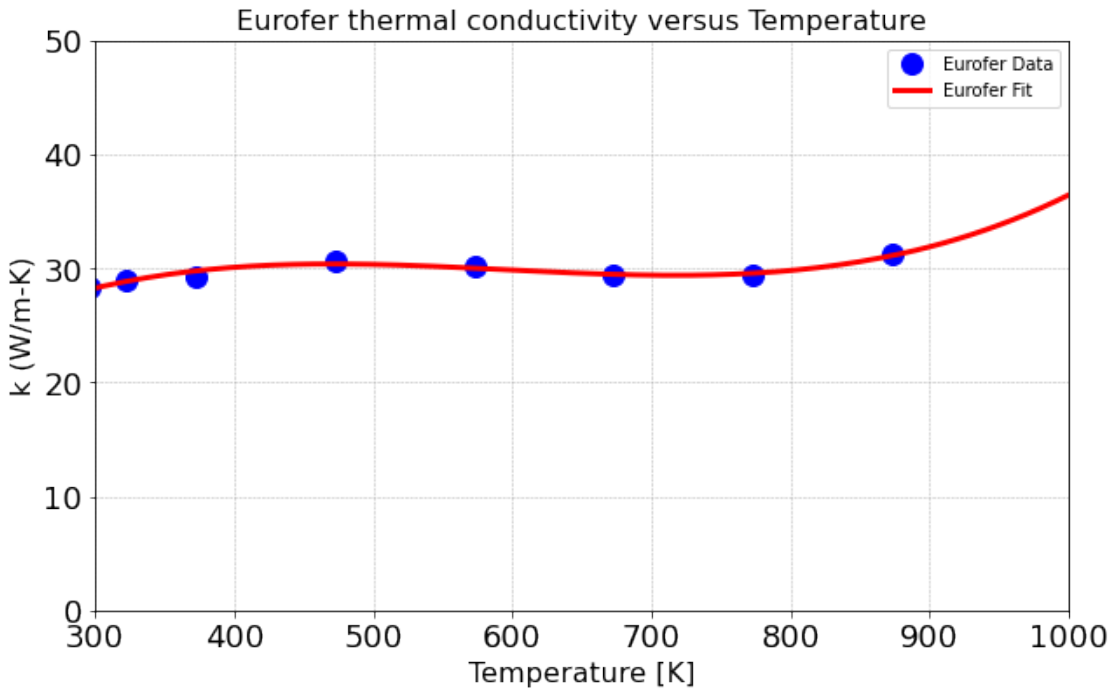
[ ]:

[3]:
```
#-1 Load the data from the Excel file
col_names = ["Temp (K)", "Thermal Conductivity, k (W/m-K)"]
df =pd.read_excel('Eurofer_physical_data.
 ↪xlsx',sheet_name="Eurofer_conductivity", header=2, usecols=[0,1], nrows=8)
T,Property= [df[col].dropna().to_numpy() for col in df]

#-2 Call the function and get the output and optimized parameters
fit_output, popt = fit_poly(T, Property,3)

#-3 Display fitting parameters and LaTeX equation for Eurofer 97
display_fitting('Eurofer 97', 'k', fit_output, popt)

#-4 Generate fitted data for Eurofer 97
T_fit = np.linspace(0, 1000, 200)
Property_fit = polynomial_model(T_fit, *popt)

#-5 Plot the data
custom_plot(T, Property, T_fit, Property_fit,
            x_label='Temperature [K]', y_label='k (W/m-K)',
            title='Eurofer thermal conductivity versus Temperature',
            data_label='Eurofer Data', fit_label='Eurofer Fit',
            scale='linear', xlim=(300, 1000), ylim=(0, 50))
```

```
<IPython.core.display.HTML object>
```

Fitting parameters for Eurofer 97

```
# Goodness-of-fit parameters

Optimized parameters: [ 5.35406912e+00  1.36128977e-01 -2.39594775e-04
1.34546275e-07]

R-squared: 0.9147117648179715

Reduced chi-squared: 0.1336821637784682

The equation for Eurofer 97 k is:


<IPython.core.display.HTML object>
```



[ ]:

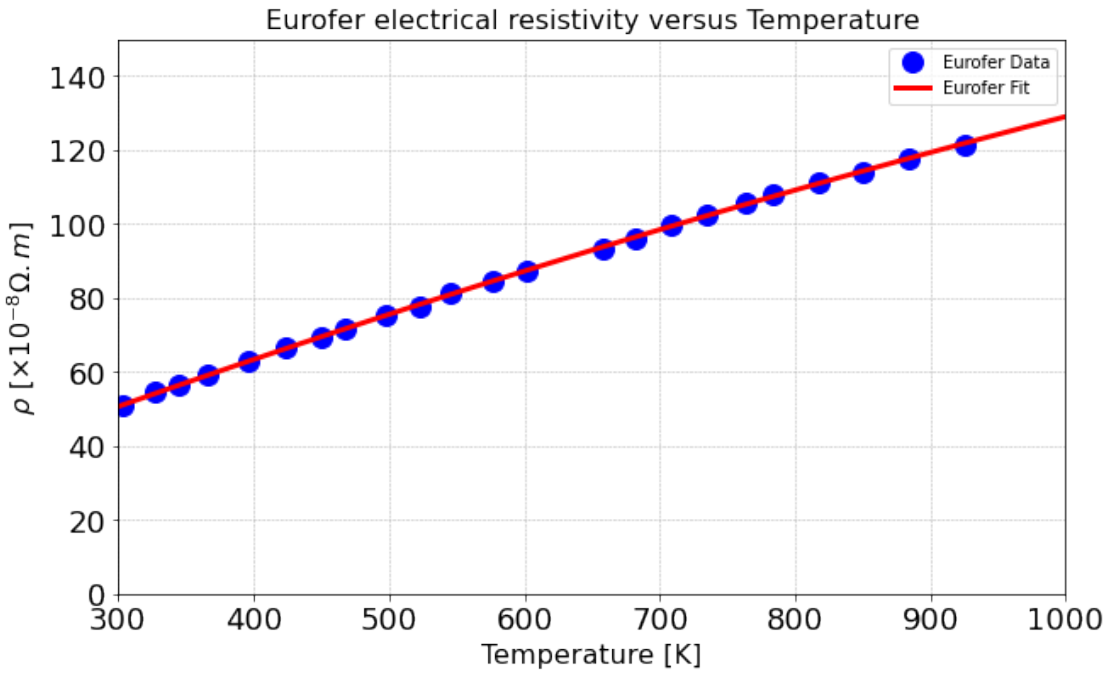
[4]:
```
#-1 Load the data from the Excel file
col_names = ["Temp (K)", "Electrical Resistivity (x10^(-8) ohm-m)"]
df =pd.read_excel('Eurofer_physical_data.
  ↪xlsx',sheet_name="Eurofer_resistivity", header=2, usecols=[0,1], nrows=23)
```

```
T,Property= [df[col].dropna().to_numpy() for col in df]

#-2 Call the function and get the output and optimized parameters
fit_output, popt = fit_poly(T, Property,2)

#-3 Display fitting parameters and LaTeX equation for Eurofer 97
# display_fitting('Eurofer 97', '\rho [10^{-8} ohm.m]', fit_output, popt)
display_fitting('Eurofer 97', r'\rho [10^{-8} \, \Omega \cdot \text{m}]',␣
 ↪fit_output, popt)

#-4 Generate fitted data for Eurofer 97
T_fit = np.linspace(0, 1000, 200)
Property_fit = polynomial_model(T_fit, *popt)

#-5 Plot the data
custom_plot(T, Property, T_fit, Property_fit,
            x_label='Temperature [K]', y_label= r'$\rho~ [\times 10^{-8} \Omega.
 ↪m]$',
            title='Eurofer electrical resistivity versus Temperature',
            data_label='Eurofer Data', fit_label='Eurofer Fit',
            scale='linear', xlim=(300, 1000), ylim=(0, 150))
```

```
<IPython.core.display.HTML object>
```

Fitting parameters for Eurofer 97

```
# Goodness-of-fit parameters

Optimized parameters: [ 9.52470861e+00  1.44689661e-01 -2.52168244e-05]

R-squared: 0.9997614573025339

Reduced chi-squared: 0.1262567312092907

The equation for Eurofer 97 \rho [10^{-8} \, \Omega \cdot \text{m}] is:


<IPython.core.display.HTML object>
```



[ ]:

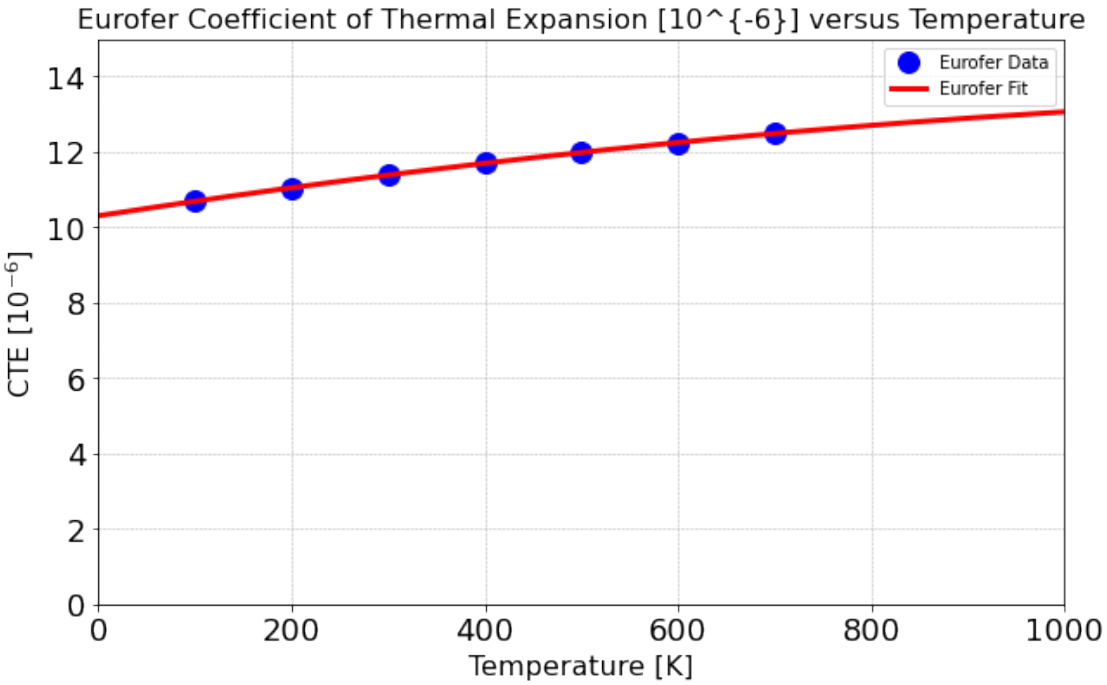
[8]:
```
#-1 Load the data from the Excel file
col_names = ["Temp (K)", "Coefficient of Thermal Expansion"]
df =pd.read_excel('Eurofer_physical_data.xlsx',sheet_name="Eurofer_CTE",␣
 ↪header=2, usecols=[0,1], nrows=10)
T,Property= [df[col].dropna().to_numpy() for col in df]

#-2 Call the function and get the output and optimized parameters
fit_output, popt = fit_poly(T, Property,2)

#-3 Display fitting parameters and LaTeX equation for Eurofer 97
# display_fitting('Eurofer 97', '\rho [10^{-8} ohm.m]', fit_output, popt)
display_fitting('Eurofer 97', r'CTE [10^{-6}]', fit_output, popt)

#-4 Generate fitted data for Eurofer 97
T_fit = np.linspace(0, 1000, 200)
Property_fit = polynomial_model(T_fit, *popt)

#-5 Plot the data
custom_plot(T, Property, T_fit, Property_fit,
            x_label='Temperature [K]', y_label= r'CTE$~[10^{-6}]$',
            title='Eurofer Coefficient of Thermal Expansion [10^{-6}] versus␣
 ↪Temperature',
            data_label='Eurofer Data', fit_label='Eurofer Fit',
```

```
            scale='linear', xlim=(0, 1000), ylim=(0, 15))
```

```
<IPython.core.display.HTML object>
```

Fitting parameters for Eurofer 97

```
# Goodness-of-fit parameters

Optimized parameters: [ 1.03028933e+01  3.97001206e-03 -1.21308018e-06]

R-squared: 0.9999871910210565

Reduced chi-squared: 8.106912048560698e-06

The equation for Eurofer 97 CTE [10^{-6}] is:

<IPython.core.display.HTML object>
```




```
[ ]:
```