Determining linear coefficients of thermal expansion from thermo-mechanical analyses (TMA) raw data

Erik Kappel[a,*]

[a]*DLR, Institute of Composite Structures and Adaptive Systems, Lilienthalplatz 7, 38108 Braunschweig, Germany*

Keywords: TMA, CTE, Radford, Process-induced distortions (PID)

Aim of the Exercise: The aim of this exercise is to extract linear coefficients of thermal expansion from real thermal-mechanical analyses data.

The essential steps are:

1. Reading the data from *.txt file
2. Determining the temperature range to examine
3. Data reduction of data of interest
4. Fitting the data with line function using Python's curve_fit module
5. Plotting raw data and the corresponding fitting result
6. Extract linear coefficient of thermal expansion
7. DONE! calculate PIDs using the provided Radford Jupyter notebook

```
#Temperature ranges you plan to evaluate
Tu = 140.0
Tl = 50.0

#Identifying the corresponding list indices
Temp_soll = ExperimentDATA[2]

upper = Temp_soll.index(Tu)
lower = Temp_soll.index(Tl)
```

Listing 1: Identifying indexes in a list belonging to a specific value in the list

Do the curve fitting with a self-defined function Extracting only the relevant data

```

time = ExperimentDATA[0][lower:upper]
...
...
...
```

Listing 2: Selecting only a specific range of the data based on indices

Do the curve fitting with a self-defined function

*Tel.:+49 531 295 2398; fax. +49 531 295 2232
Email address: erik.kappel@dlr.de (Erik Kappel)

```
#Your function used for fitting
def func(x, m, b):
    return m * x + b

#Parametric quadratic function
#def funcquad(x, a, b ,c ):
#    return a * x**2 + b*x + c

#That's all you need for simple curve-fitting tasks
from scipy.optimize import curve_fit

x  = ?
y  = ?

popt, pcov = curve_fit(func, x, y)
#popt[0] refers to m of func and popt[1] refers to b

CTE = popt[0]*1.0e6 # multiplied with 1.0E6 to have ppm/K value
Fittedeps = func(np.asarray(x),*popt)

print('The CTE is %4.2f ppm/K' % (CTE))
```

Listing 3: Curve fitting framework

```
plt.plot(ExperimentDATA[0],ExperimentDATA[4],'r--')
plt.plot(time,Fittedeps,'b',linewidth=2.0)

middleindex = int(len(Fittedeps) / 2) - 1

plt.annotate('lin. CTE = %4.2f ppm/K'%(CTE), xy=(time[middleindex], Fittedeps[middleindex
    ]), xytext=(time[middleindex], 0.15*Fittedeps[middleindex]),
            arrowprops=dict(facecolor='black', shrink=0.00),
            )

plt.xlabel('time [s]')
plt.ylabel('strain [-]')
plt.show(True)
```

Listing 4: Curve fitting framework

Create a picture...

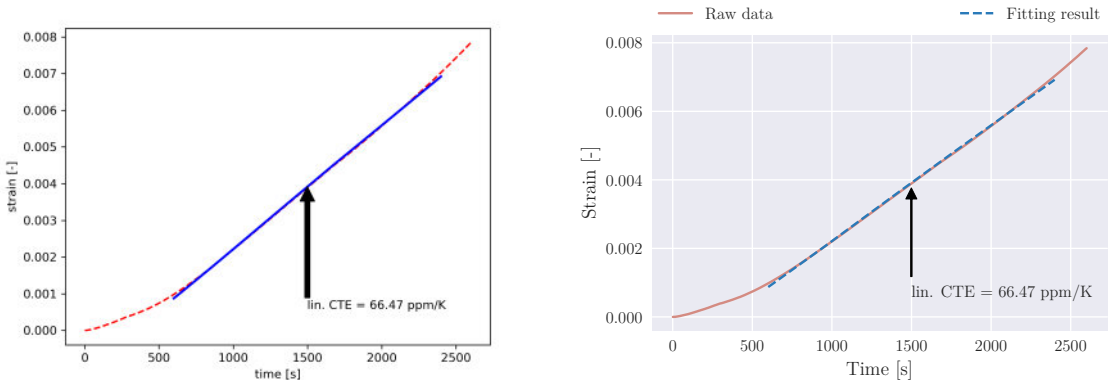

Figure 1: Preliminary and final picture

```
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib import rc

Resolution = 300
fontsizes = [8, 10, 12 ,18]

# Figure dimensions
Width_in_cm = 15
fig = plt.figure(figsize=(Width_in_cm/2.54 , Width_in_cm*(2/3)/2.54))

#Define axes handle
ax = fig.add_subplot(111)
ax.tick_params(axis='both', which='major', labelsize=fontsizes[2])
ax.tick_params(axis='both', which='minor', labelsize=fontsizes[2])

################
sns.set(style="darkgrid")

rc('text.latex',preamble=r'\usepackage[T1]{fontenc},\usepackage{amsmath},\usepackage{
    lmodern}')
rc('text', usetex=True)
rc('font', size= fontsizes[2])
rc('font',**{'family':'serif','serif':['DejaVu Serif']})
################


REDS = ['#E24B56','#BC6767','#D48B82','#ECD4CC']
BLUES = ['#DAE8F5','#BAD6EA','#88BEDC','#539DCC','#2A7AB9','#0B559F']
GREENS = ['#C4E4CC','#91C9AB','#64A698','#427F82','#2E5365','#19253D']
```

```
#plt.plot(ExperimentDATA[0],ExperimentDATA[4],'r--')
#plt.plot(time,Fittedeps,'b',linewidth=2.0)

line1, = ax.plot(ExperimentDATA[0],ExperimentDATA[4], '-', color=REDS[2], label=r'Raw
    data')
line2, = ax.plot(time,Fittedeps, '--', color=BLUES[4], label=r'Fitting result')


ax.annotate('lin. CTE = %4.2f ppm/K'%(CTE), xy=(time[middleindex], Fittedeps[middleindex
    ]), xytext=(time[middleindex], 0.15*Fittedeps[middleindex]),
           arrowprops=dict(facecolor='black', shrink=0.05,width=1.0,headwidth=8,
               headlength=8),
           )
lines = [line1,line2]
ax.legend(lines, [l.get_label() for l in lines],bbox_to_anchor=(0., 1.02, 1., .102), loc
    =3, ncol=2, mode="expand", borderaxespad=0.,fontsize=12)

ax.grid(True)
ax.set_xlabel(r'Time [s]',fontsize=14)
ax.set_ylabel(r'Strain [-]',fontsize=14)


plt.show()

Outputname = "Output"

name = Outputname +'.pdf'
fig.savefig(name, format='PDF', dpi=Resolution, bbox_inches='tight')#, ,
    bbox_extra_artists=(leg,)
name = Outputname +'.png'
#fig.savefig(name, format='PNG', dpi=Resolution, bbox_inches='tight')
```

Listing 5: Code for 'more-professional picture'