Import the required libraries

In [1]:
```
%matplotlib widget
import hyperspy.api as hs
import numpy as np
import matplotlib.pyplot as plt
import atomap.api as am
import tkinter as tk
from tkinter.filedialog import askdirectory
import os
hs.preferences.GUIs.warn_if_guis_are_missing = False
hs.preferences.save()
plt.rcParams['figure.figsize'] = (8,8)
```

Normalization of the intensity profile

- Loading the atom lattice:

In [2]:
```
root = tk.Tk()
root.attributes('-topmost',True)
root.iconify()
file_path = askdirectory(parent=root)
root.destroy()
atom_lattice = am.load_atom_lattice_from_hdf5(file_path+'\\data.hdf5',construct_zone_axes=False)
sublattice_A=atom_lattice.sublattice_list[0]
sublattice_B=atom_lattice.sublattice_list[1]
atom_lattice.units=atom_lattice.sublattice_list[0].units
atom_lattice.pixel_size=atom_lattice.sublattice_list[0].pixel_size
```

> `text` ; selected intensity map, for example:

- `im_A_high_pass_rl_imag` for A intensity map of the RL deconvolution with high-pass image.
- `im_B_band_pass_pca_imag` for B intensity map of the PCA with band-pass image.
- ...

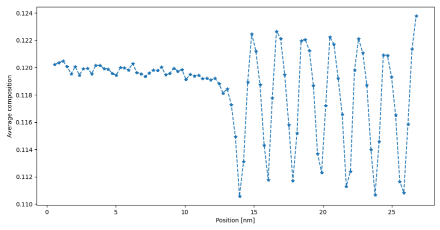
In [3]:
```
text='im_A_band_pass_pca_imag'

intensity_map=np.load(file_path+'\\'+text+'.npy')
avg_intensity=np.nanmean(intensity_map[:,:,0],axis=1)
std_dev_intensity=np.nanstd(intensity_map[:,:,0],axis=1)
avg_axis=np.nanmean(intensity_map[:,:,2],axis=1)*atom_lattice.pixel_size
nominal_composition=1
muraki_array=[]
muraki_values=[]
```

- Intensity profile of the selected intensity map:

In [4]:
```
plt.figure(figsize=(12,6))
plt.plot(avg_axis,avg_intensity,'*--')
plt.xlabel('Position ['+atom_lattice.units+']')
plt.ylabel('Average composition')
plt.show()
```

Figure



Histogram to obtain:

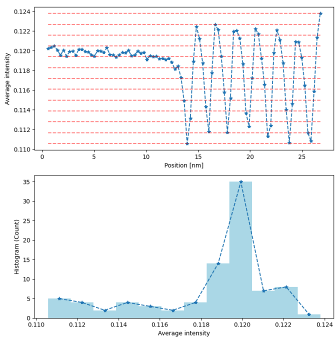
- Mean value of the barriers
- Mean value of the quantum well

In [5]:
```
from scipy.stats import binned_statistic
from scipy.signal import find_peaks

count_binned=binned_statistic(avg_intensity,avg_intensity, 'count', bins=12)
bin_centers=(count_binned[1][1:] + count_binned[1][:-1])/2
mean_binned=binned_statistic(avg_intensity,avg_intensity, 'mean', bins=12)
pos_peaks, _ = find_peaks(count_binned[0], height=0)
pos_peaks_sorted=pos_peaks[np.argsort(count_binned[0][pos_peaks])]
peaks_sorted=mean_binned[0][pos_peaks_sorted]

fig, (ax1, ax2) = plt.subplots(2, 1)
ax1.plot(avg_axis,avg_intensity,'*--')
ax1.set(xlabel='Position ['+atom_lattice.units+']',ylabel='Average intensity')
for i in count_binned[1]:
    ax1.plot(avg_axis,[i]*len(avg_intensity),linestyle='--',color='red',alpha=0.5)
_, _, patches = ax2.hist(avg_intensity,bins=count_binned[1],color='lightblue')
ax2.plot(bin_centers,count_binned[0],'*--')
ax2.set(xlabel='Average intensity',ylabel='Histogram (Count)')
fig.tight_layout()
plt.show()
```

Figure



> **n_lower_limit**: bin index for the lower limit
> **n_upper_limit**: bin index for the upper limit

In [6]:
```
n_lower_limit=2
n_upper_limit=-4

lower_limit,upper_limit=count_binned[1][n_lower_limit],count_binned[1][n_upper_limit]
positions_l=np.where(avg_intensity<lower_limit)
i_barriers=np.nanmean(avg_intensity[positions_l])
positions_u=np.where(avg_intensity>upper_limit)
i_quantum_well=np.nanmean(avg_intensity[positions_u])
print('Mean intensity of the barriers: '+str(i_barriers))
print('Mean intensity of the quantum well: '+str(i_quantum_well))
```

```
Mean intensity of the barriers: 0.11148113484155317
Mean intensity of the quantum well: 0.12177308822412891
```

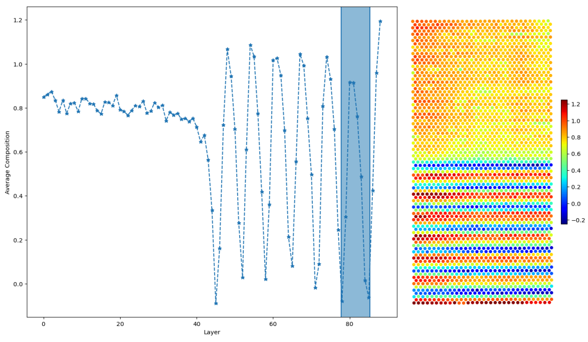
- Selection of the region with the interfaces after normalization:

In [7]:
```
import matplotlib.pyplot as plt
from matplotlib.widgets import SpanSelector
from matplotlib import gridspec

normalized_array=(intensity_map-i_barriers)/(i_quantum_well-i_barriers)
avg_norm=np.nanmean(normalized_array[:,:,0],axis=1)
std_dev_norm=np.nanstd(normalized_array[:,:,0],axis=1)

fig = plt.figure(figsize=(14, 8))
gs = gridspec.GridSpec(1, 2, width_ratios=[2, 1])
ax0 = plt.subplot(gs[0])
img1=ax0.plot(avg_norm,'*--')
ax0.set(xlabel='Layer',ylabel='Average Composition')
def onselect(xmin, xmax):
    global x_pos
    x_pos = np.array([xmin,xmax])
span = SpanSelector(
    ax0,
    onselect,
    "horizontal",
    useblit=True,
    props=dict(alpha=0.5, facecolor="tab:blue"),
    interactive=True,
    drag_from_anywhere=True
)
ax1 = plt.subplot(gs[1])
img2=ax1.scatter(intensity_map[:,:,1],intensity_map[:,:,2],s=20,c=normalized_array[:,:,0],cmap='jet',vmin=-0.25,vmax=1.25)
fig.colorbar(img2,shrink=0.4,pad=0)
ax1.axis('scaled')
ax1.axis('off')
ax1.set_ylim(ax1.get_ylim()[::-1])
plt.tight_layout()
plt.show()
```

Figure



In [X]:
```
muraki_positions=np.arange(x_pos[0]+1,x_pos[1]+1,dtype=int)
muraki_signal=avg_norm[muraki_positions]
std_dev=std_dev_norm[muraki_positions]
sc=hs.signals.Signal1D(muraki_signal)
print('Lower layer of the selection: '+str(muraki_positions[0]))
print('Upper layer of the selection: '+str(muraki_positions[-1]))
```

```
Lower layer of the selection: 78
Upper layer of the selection: 85
```

Fitting (one segregation coefficient):

$$x(n) = \begin{cases} x_0(1-S^n): & 1 \leq n \leq N \\ x_0(1-S^N)S^{n-N}: & n > N \end{cases}$$

- Build of the Muraki model for one segregation coefficient:

In [X]:
```
from hyperspy.component import Component
class Muraki(Component):
    def __init__(self, parameter_1=1, parameter_2=2, parameter_3=3):
        Component.__init__(self, ('x0', 's', 'N'))
        self.x0.value = 1.5
        self.s.value = 0.5
        self.N.value = 5
        self.x0.bmin = 0
        self.x0.bmax = 1
        self.s.bmin = 0
        self.s.bmax = 1
        self.N.bmin = 0
        self.N.bmax = 50
    def function(self, x):
        x0 = self.x0.value
        s = self.s.value
        N = self.N.value
        return np.piecewise(x,[((x >= 1.0) & (x<= N)),x >= N],[lambda x : x0*(1.0 -s**x), lambda x: x0*(1 -s**x)*s**(x-N)])
```

In [X]:
```
muraki_model = sc.create_model()
muraki = Muraki()
muraki_model.append(muraki)
muraki_model.fit()
muraki_model.print_current_values()
muraki1_values=muraki.__dict__
```

Model1D:

current_component_values: Muraki

Active: True

Parameter Name	Free	Value	Std	Min	Max	Linear
x0	True	0.855653	0.141334	0	1	False
s	True	0.35202	0.204317	0	1	False
N	True	4.34022	0.660562	0	50	False

- Plot of the fitted model and the input data:

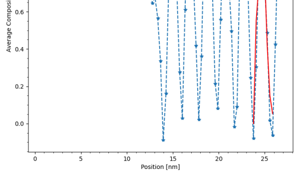
In [X]:
```
from sklearn.metrics import r2_score

def f(x,x0,s,N):
    return np.piecewise(x,[((x >= 1.0) & (x<= N)),x >= N],[lambda x : x0*(1.0 -s**x), lambda x: x0*(1 -s**x)*s**(x-N)])
x=np.arange(0,len(sc.data),dtype=float)
y_pred=f(x,muraki.x0.value,muraki.s.value,muraki.N.value)
r2_parameter=r2_score(sc.data[0::], y_pred[0::])

from matplotlib.offsetbox import AnchoredText

plt.figure()
values1={'x':x,'muraki_positions':muraki_positions,'y_pred':y_pred,'r2_parameter':r2_parameter}
plt.plot(avg_axis[np.arange(0,len(avg_intensity))],avg_norm,'*--')
plt.plot(avg_axis[x.astype(int)+muraki_positions[0]],y_pred,'-',color='red')
plt.xlabel('Position [nm]')
plt.ylabel('Average Composition')
plt.minorticks_on()
plot=plt.gca()
label='$R^2 = $'+str(np.round(r2_parameter,3))
at = AnchoredText(label, prop=dict(size=10), frameon=True, loc='upper right')
at.patch.set(edgecolor='lightgray')
at.patch.set_boxstyle('round,pad=0.,rounding_size=0.1')
plot.add_artist(at)
plt.show()
```

Figure



Fitting (two segregation coefficients):

$$x(n) = \begin{cases} x_0(1-S_l^n): & 1 \leq n \leq N \\ x_0(1-S_l^N)S_u^{n-N}: & n > N \end{cases}$$

- Build of the Muraki model for two segregation coefficients:

In [X]:
```
class Muraki2(Component):
    def __init__(self, parameter_1=1, parameter_2=2, parameter_3=3, parameter_4=4):
        Component.__init__(self, ('x0', 's1', 's2', 'N'))
        self.x0.value = 1.25
        self.s1.value = 0.5
        self.s2.value = 0.5
        self.N.value = 4
        self.x0.bmin = 0
        self.x0.bmax = 1
        self.s1.bmin = 0
        self.s1.bmax = 1
        self.s2.bmin = 0
        self.s2.bmax = 1
        self.N.bmin = 0
        self.N.bmax = 50
    def function(self, x):
        x0 = self.x0.value
        s1 = self.s1.value
        s2 = self.s2.value
        N = self.N.value
        return np.piecewise(x,[((x >= 1.0) & (x<= N)),x >= N],[lambda x : x0*(1.0 -s1**x), lambda x: x0*(1 -s2**x)*s2**(x-N)])
```

In [X]:
```
muraki_model = sc.create_model()
muraki = Muraki2()
muraki_model.append(muraki)
muraki_model.fit()
muraki_model.print_current_values()
muraki2_values=muraki.__dict__
```

Model1D:

current_component_values: Muraki2

Active: True

Parameter Name	Free	Value	Std	Min	Max	Linear
x0	True	0.944393	0.224959	0	1	False
s1	True	0.483299	0.233872	0	1	False
s2	True	0.0287033	0.329012	0	1	False
N	True	4.81455	0.612154	0	50	False

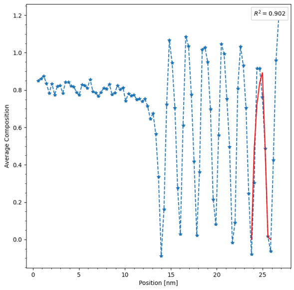
- Plot of the fitted model and the input data:

In [X]:

```
def f(x,x0,s1,s2,N):
    return np.piecewise(x,[((x >= 1.0) & (x<= N)),x >= N],[lambda x : x0*(1.0 -s1**x), lambda x: x0*(1 -s2**x)*s2**(x-N)])
x=np.arange(0,len(sc.data),dtype=float)
y_pred=f(x,muraki.x0.value,muraki.s1.value,muraki.s2.value,muraki.N.value)
r2_parameter=r2_score(sc.data[1::], y_pred[1::])

plt.figure()
values2={'x':x,'muraki_positions':muraki_positions,'y_pred':y_pred,'r2_parameter':r2_parameter}
plt.plot(avg_axis[np.arange(0,len(avg_intensity))],avg_norm,'*--')
plt.plot(avg_axis[x.astype(int)+muraki_positions[0]],y_pred,'-',color='red')
plt.xlabel('Position [nm]')
plt.ylabel('Average Composition')
plt.minorticks_on()
plot=plt.gca()
label='$R^2 = $'+str(np.round(r2_parameter,3))
at = AnchoredText(label, prop=dict(size=10), frameon=True, loc='upper right')
at.patch.set(edgecolor='lightgray')
at.patch.set_boxstyle('round,pad=0.,rounding_size=0.1')
plot.add_artist(at)
plt.show()
```

Figure



- Add the values to a single list to fit the next peak.

In [X]:

```
muraki_array.append([muraki1_values,muraki2_values])
muraki_values.append([values1,values2])
```

Plot all the fits

- Save to a .csv file

In [101…

```
muraki_array=np.asarray(muraki_array)
muraki_values=np.asarray(muraki_values)
matrix = np.empty(shape=muraki_array.shape, dtype='object')
for i,j in [(x,y) for x in range(muraki_array.shape[0]) for y in range(muraki_array.shape[1])]:
    muraki_dict={}
    for k in muraki_array[i,j]['parameters']:
        muraki_dict[k.name]=(muraki_array[i,j][k.name].value,muraki_array[i,j][k.name].std)
        muraki_dict['r2_parameter']=muraki_values[i,j]['r2_parameter']
    matrix[i,j]=muraki_dict

import pandas as pd

fringes=['F'+k for k in np.arange(1,matrix.shape[0]+1).astype(str)]
df = pd.DataFrame(matrix, columns = ['S1','S1 and S2'], index = fringes)
df.to_csv(file_path+'\\muraki_values.csv')

r2_sum1=0
for i in np.arange(matrix.shape[0]):
    r2_sum1=r2_sum1+matrix[i,0]['r2_parameter']
r2_sum1=r2_sum1/matrix.shape[0]
r2_sum2=0
for i in np.arange(matrix.shape[0]):
    r2_sum2=r2_sum2+matrix[i,1]['r2_parameter']
r2_sum2=r2_sum2/matrix.shape[0]
```

- One segregation coefficient:

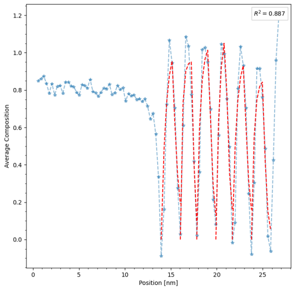
In [102…

```
plt.figure()
plt.plot(avg_axis[np.arange(0,len(avg_intensity))],avg_norm,'*--',alpha=0.5)
s_model=1
for i in range (len(muraki_values)):
    plt.plot(avg_axis[muraki_values[i][s_model-1]['x'].astype(int)+muraki_values[i][s_model-1]['muraki_positions'][0]],muraki_values[i][s_model-1]['y_pred'],'--',color='red')

from matplotlib.offsetbox import AnchoredText

plt.xlabel('Position [nm]')
plt.ylabel('Average Composition')
plt.minorticks_on()
plot=plt.gca()
label='$R^2 = $'+str(np.round(r2_sum1,3))
at = AnchoredText(label, prop=dict(size=10), frameon=True, loc='upper right')
at.patch.set(edgecolor='lightgray')
at.patch.set_boxstyle('round,pad=0.,rounding_size=0.1')
plot.add_artist(at)
plt.show()
```

Figure



In [103…

```
plt.savefig(file_path+'\\muraki_1s_'+text+'.png',dpi=300,transparent=True,bbox_inches='tight')
```

- Two segregation coefficients:

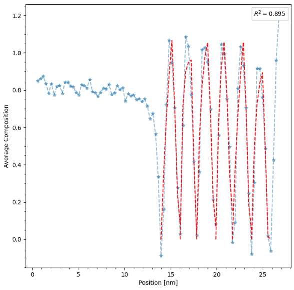
In [104…

```
plt.figure()
plt.plot(avg_axis[np.arange(0,len(avg_intensity))],avg_norm,'*--',alpha=0.5)
s_model=2
for i in range (len(muraki_values)):
    plt.plot(avg_axis[muraki_values[i][s_model-1]['x'].astype(int)+muraki_values[i][s_model-1]['muraki_positions'][0]],muraki_values[i][s_model-1]['y_pred'],'--',color='red')

from matplotlib.offsetbox import AnchoredText

plt.xlabel('Position [nm]')
plt.ylabel('Average Composition')
plt.minorticks_on()
plot=plt.gca()
label='$R^2 = $'+str(np.round(r2_sum2,3))
at = AnchoredText(label, prop=dict(size=10), frameon=True, loc='upper right')
at.patch.set(edgecolor='lightgray')
at.patch.set_boxstyle('round,pad=0.,rounding_size=0.1')
plot.add_artist(at)
plt.show()
```

Figure



In [105…

```
plt.savefig(file_path+'\\muraki_2s_'+text+'.png',dpi=300,transparent=True,bbox_inches='tight')
```