

Notebook-BHO-Size

March 13, 2022

1 Evaluate the size of BaHfO₃ measured by manual tracing in Fiji

Author: Lukas Grünewald (ORCID: <https://orcid.org/0000-0002-5898-0713>)

Supplementary information to the paper: “**Microstructure and pinning properties of CSD-grown SmBa₂Cu₃O_{7-δ} films with and without BaHfO₃ nanoparticles**”

The projected BaHfO₃ (BHO) size was measured from STEM images in Fiji using manual labeling with the *Labkit* plugin (see <https://imagej.net/plugins/labkit>).

M. Arzt, J. Deschamps, C. Schmied, T. Pietzsch, D. Schmidt, P. Tomancak, R. Haase, F. Jug, *Frontiers in Computer Science* **2022**, 4. DOI: [10.3389/fcomp.2022.777728](https://doi.org/10.3389/fcomp.2022.777728)

The output .csv files contain the measured particle metrics.

This notebook can be run step-by-step using SHIFT+ENTER or using Run All Cells from the menu.

1.1 Preparation

```
[1]: #Use watermark package to document package versions, https://github.com/rasbt/
      ↪ watermark
      %load_ext watermark
```

```
[2]: import pandas as pd
      import numpy as np
      import matplotlib.pyplot as plt
      import glob
```

```
[3]: from scipy.optimize import curve_fit
      #For error propagation:
      from uncertainties import ufloat, unumpy
      from uncertainties.umath import *
```

```
[4]: #Print package versions
      %watermark -i -v -u -m --iversions
```

Last updated: 2022-03-13T18:33:44.177078+01:00

Python implementation: CPython
Python version : 3.9.6
IPython version : 7.26.0

Compiler : MSC v.1916 64 bit (AMD64)
 OS : Windows
 Release : 10
 Machine : AMD64
 Processor : Intel64 Family 6 Model 58 Stepping 9, GenuineIntel
 CPU cores : 4
 Architecture: 64bit

numpy : 1.22.3
 pandas : 1.3.1
 matplotlib : 3.4.2
 uncertainties: 3.1.6

1.1.1 Define log-normal distribution

```
[5]: def pdf(x, mu, sigma):
      """Probability density function of a log-normal distribution"""
      return (np.exp(-(np.log(x) - mu)**2 / (2 * sigma**2)) / (x * sigma * np.
      ↪sqrt(2 * np.pi)))
```

For details and a discussion about log-normal distributions, see

E. Limpert, W. A. Stahel, M. Abbt, *BioScience* **2001**, 51, 341. DOI: [10.1641/0006-3568\(2001\)051](https://doi.org/10.1641/0006-3568(2001)051) or Wikipedia (https://en.wikipedia.org/wiki/Log-normal_distribution):

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma x} \exp\left(-\frac{(\ln(x) - \mu)^2}{2\sigma^2}\right), \quad x > 0$$

with the mean μ and standard deviation σ of $\ln(x)$ as the fit parameters.

1.1.2 Mode M , arithmetic mean μ_a , and standard deviation σ_a of the values x (i.e., the measured values):

Fitting errors on μ and σ will be propagated by the [uncertainties package](#).

$$M(x) = e^{\mu - \sigma^2}$$

$$\mu_a(x) = e^{\mu + \frac{\sigma^2}{2}}$$

$$\sigma_a(x) = e^{\mu + \frac{1}{2}\sigma^2} \sqrt{e^{\sigma^2} - 1}$$

1.1.3 Scatter intervals

For a log-normal distribution, with geometric mean and standard deviation $\mu^* = e^\mu$ and $\sigma^* = e^\sigma$,

$$[\mu^*/\sigma^*, \mu^* \cdot \sigma^*]$$

contains 68 %, and

$$[\mu^*/(\sigma^*)^2, \mu^* \cdot (\sigma^*)^2]$$

contains 95 % of the probability.

1.2 BHO-nanoparticle size in SmBCO

All nanoparticles (in the film, at the film-substrate surface, and the film's surface) are evaluated.

```
[6]: #Grab and stack all csv files
df = pd.concat([pd.read_csv(filename, header=0, index_col=None) for filename in
↳glob.glob('*.csv')], ignore_index=True)
```

```
[7]: df.shape
```

```
[7]: (121, 35)
```

```
[8]: df.head()
```

```
[8]:
```

		Area	Mean	StdDev	Mode	Min	Max	X	Y	\
0	1	628.46583	255	0	255	255	255	150.92276	311.16138	
1	2	355.50003	255	0	255	255	255	1508.71700	308.69204	
2	3	239.70265	255	0	255	255	255	1661.28346	308.64440	
3	4	155.08948	255	0	255	255	255	837.30654	314.61062	
4	5	199.16317	255	0	255	255	255	1071.85114	313.75590	

		XM	...	%Area	RawIntDen	Slice	FeretX	FeretY	FeretAngle	\
0		150.92276	...	100	770865	1	293	707	38.45371	
1		1508.71700	...	100	436050	1	3275	689	13.39250	
2		1661.28346	...	100	294015	1	3617	688	6.45882	
3		837.30654	...	100	190230	1	1818	683	164.53878	
4		1071.85114	...	100	244290	1	2318	694	9.30994	

		MinFeret	AR	Round	Solidity
0		22.61465	1.77111	0.56462	0.96705
1		15.50246	1.84365	0.54240	0.96774
2		14.13460	1.49171	0.67037	0.93702
3		10.94291	2.11321	0.47321	0.87972
4		9.57505	3.29106	0.30385	0.92382

```
[5 rows x 35 columns]
```

1.2.1 Equivalent diameter

We calculate circles with equal diameter as the measured BHO particle size (Area column).

```
[9]: data = df['Area'].to_numpy()
```

Calculate equivalent diameters from areas:

```
[10]: data = np.sqrt(4*data/np.pi)
```

Fit distribution and plot:

```

[11]: fig, ax = plt.subplots(figsize=(3.167, 2.440))

n, bins, patches = ax.hist(data, bins='auto', density=True, facecolor = 'grey',
    ↪ alpha = 0.5, label=None)
centers = (0.5*(bins[1:]+bins[:-1]))

# Fit
pars, cov = curve_fit(pdf, centers, n)

# Draw pdf
distc = 'k'
xmin, xmax = ax.get_xlim()
x = np.linspace(xmin, xmax, 1000)
ax.plot(x, pdf(x, *pars), distc, linewidth = 1.5, label=None)

# Add fit parameters (mu, sigma, mode) as labels
# Errors are from diagonal elements of cov (covariance matrix) -->
    ↪ sqrt(Var)=Std dev
mu, sigma = unumpy.uarray(pars, np.sqrt(np.diag(cov)))

muStar = exp(mu)
sigStar = exp(sigma)
confidence68 = (muStar/sigStar, muStar*sigStar)
confidence95 = (muStar/sigStar**2, muStar*sigStar**2)

std = exp(mu + sigma**2/2) * sqrt(exp(sigma**2)-1)
mode = exp(mu - sigma ** 2)
mean = exp(mu + sigma ** 2 / 2)

print(f'Mode:\t\t {mode}')
print(f'Mean:\t\t {mean}')
print(f'Std. dev.:\t {std}')
print(f'68% conf. intervall: {confidence68}')
print(f'95% conf. intervall: {confidence95}')

lb_std = rf'$\sigma_{\mathrm{{a}}} = {np.round(std.nominal_value,1)}$ nm'
lb_mean = rf'$\mu_{\mathrm{{a}}} = {np.round(mean.nominal_value,1)}$ nm'
lb_m = rf'$m = {np.round(mode.nominal_value,1)}$ nm'
lb_mu = rf'$\mu = {np.round(mu.nominal_value,3)} \pm {np.round(mu.std_dev,3)}$'
lb_sig = rf'$\sigma = {np.round(sigma.nominal_value,3)} \pm {np.round(sigma.
    ↪ std_dev,3)}$'

#Mode
ax.plot([mode.nominal_value, mode.nominal_value], [0, pdf(mode.nominal_value,
    ↪ *pars)], c='tab:orange', ls='--', label=lb_m)

#Mean

```

```

ax.plot([mean.nominal_value, mean.nominal_value], [0, pdf(mean.nominal_value,
↳*pars)], c='tab:blue', ls='-.', label=lb_mean)

#Confidence interval
ci = confidence68
ax.plot([ci[0].nominal_value, ci[0].nominal_value], [0, pdf(ci[0].
↳nominal_value, *pars)], c=distc, ls='dotted')
ax.plot([ci[1].nominal_value, ci[1].nominal_value], [0, pdf(ci[1].
↳nominal_value, *pars)], c=distc, ls='dotted')

#Dummy plots for labels
plt.plot([], [], ' ', label=lb_std)
plt.plot([], [], ' ', label='Fit parameters:')
plt.plot([], [], ' ', label=lb_mu)
plt.plot([], [], ' ', label=lb_sig)

#Cosmetics
#ax.set_title(f'Size of BaHfO$_3$ nanoparticles, # particles: {data.shape[0]}',
↳fontsize=9)
ax.set_title(f'Size of BaHfO$_3$ nanoparticles', fontsize=9)
ax.set_xlabel('Equivalent diameter (nm)', fontsize=9)
ax.set_ylabel("Normalized frequency", fontsize=9)
plt.tick_params(axis='both', which='major', labelsize=7.5)
#ax.set_xlim(0, 40)
ax.set_ylim(0, 0.12)

ax.legend(loc='upper center', handlelength=1.5, fontsize=7.8, ncol=2,
↳columnspacing=0.1)

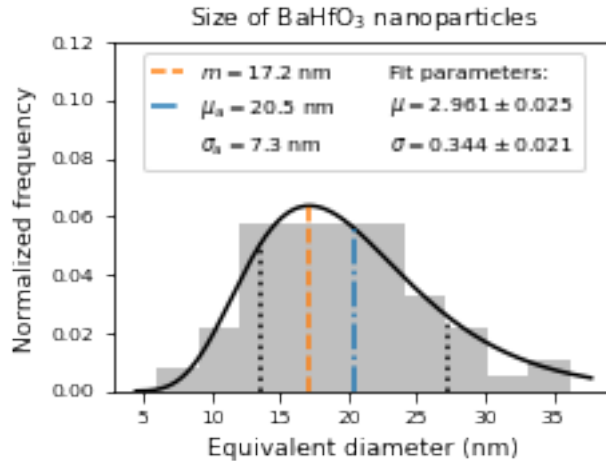
plt.tight_layout(pad=0.2)
fig.savefig('BHO_size.pdf', pad_inches=0.05)

```

```

Mode:          17.2+/-0.5
Mean:          20.5+/-0.5
Std. dev.:     7.3+/-0.5
68% conf. intervall: (13.691451779098589+/-0.4428345738544329,
27.23023928128246+/-0.8807313937656698)
95% conf. intervall: (9.708419255316334+/-0.46828087587488765,
38.401875552380766+/-1.8522957698865212)

```



1.3 BHO-nanoparticle size in the SmBCO film

Only with BHO particles in the film (ignore BHO particles (i) at the substrate interface and (ii) at the surface)

```
[12]: #Grab and stack all csv files
df = pd.read_csv('BHO_layer.csv', header=0, index_col=None)
```

```
[13]: data = df['Area'].to_numpy()
```

Calculate equivalent diameters from areas:

```
[14]: data = np.sqrt(4*data/np.pi)
```

```
[15]: fig, ax = plt.subplots(figsize=(3.3, 2.440))

n, bins, patches = ax.hist(data, bins='auto', density=True, facecolor = 'grey',
    ↪alpha = 0.5, label=None)
centers = (0.5*(bins[1:]+bins[:-1]))

# Fit
pars, cov = curve_fit(pdf, centers, n)

# Draw pdf
distc = 'k'
xmin, xmax = ax.get_xlim()
x = np.linspace(xmin, xmax, 1000)
ax.plot(x, pdf(x, *pars), distc, linewidth = 1.5, label=None)

# Add fit parameters (mu, sigma, mode) as labels
```

```

# Errors are from diagonal elements of cov (covariance matrix) -->
    ↪ sqrt(Var)=Std dev
mu, sigma = unumpy.uarray(pars, np.sqrt(np.diag(cov)))

muStar = exp(mu)
sigStar = exp(sigma)
confidence68 = (muStar/sigStar, muStar*sigStar)
confidence95 = (muStar/sigStar**2, muStar*sigStar**2)

std = exp(mu + sigma**2/2) * sqrt(exp(sigma**2)-1)
mode = exp(mu - sigma ** 2)
mean = exp(mu + sigma ** 2 / 2)

print(f'Mode:\t\t {mode}')
print(f'Mean:\t\t {mean}')
print(f'Std. dev.:\t {std}')
print(f'68% conf. intervall: {confidence68}')
print(f'95% conf. intervall: {confidence95}')

lb_std = rf'$\sigma_{\mathregular{{a}}} = {np.round(std.nominal_value,1)}$ nm'
lb_mean = rf'$\mu_{\mathregular{{a}}} = {np.round(mean.nominal_value,1)}$ nm'
lb_m = rf'$m = {np.round(mode.nominal_value,1)}$ nm'
lb_mu = rf'$\mu = {np.round(mu.nominal_value,3)} \pm {np.round(mu.std_dev,3)}$'
lb_sig = rf'$\sigma = {np.round(sigma.nominal_value,3)} \pm {np.round(sigma.
    ↪ std_dev,3)}$'

#Mode
ax.plot([mode.nominal_value, mode.nominal_value], [0, pdf(mode.nominal_value,
    ↪ *pars)], c='tab:orange', ls='--', label=lb_m)

#Mean
ax.plot([mean.nominal_value, mean.nominal_value], [0, pdf(mean.nominal_value,
    ↪ *pars)], c='tab:blue', ls='-.', label=lb_mean)

#Confidence interval
ci = confidence68
ax.plot([ci[0].nominal_value, ci[0].nominal_value], [0, pdf(ci[0].
    ↪ nominal_value, *pars)], c=distc, ls='dotted')
ax.plot([ci[1].nominal_value, ci[1].nominal_value], [0, pdf(ci[1].
    ↪ nominal_value, *pars)], c=distc, ls='dotted')

#Dummy plots for labels
plt.plot([], [], ' ', label=lb_std)
plt.plot([], [], ' ', label='Fit parameters:')
plt.plot([], [], ' ', label=lb_mu)
plt.plot([], [], ' ', label=lb_sig)

```

```

#Cosmetics
ax.set_title(f'Size of BaHfO3 nanoparticles', fontsize=9)
ax.set_xlabel('Equivalent diameter (nm)', fontsize=9)
ax.set_ylabel("Normalized frequency", fontsize=9)
plt.tick_params(axis='both', which='major', labelsize=7.5)
ax.set_ylim(0, 0.15)

ax.legend(loc='best', handlelength=1.2, fontsize=9, ncol=2, columnspacing=0.1)
plt.tight_layout(pad=0.2)
fig.savefig('BHO_size_onlyLayer.pdf', dpi=600, pad_inches=0.05)

```

Mode: 17.1+/-0.8
 Mean: 20.0+/-0.8
 Std. dev.: 6.7+/-0.9
 68% conf. intervall: (13.696012482969874+/-0.7225865723037888,
 26.20453914419791+/-1.3825226972121238)
 95% conf. intervall: (9.901544682219235+/-0.7853357891508291,
 36.24663693876978+/-2.874882873122815)

