XRD Comparison Graph

In [443…

```
# Import libraries
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
```

XRD Data was extracted from graph. Peaks correspond to the presence of phases, with the chemical composition linked to some known database. We're looking at chemical compositions involving chromium, tungsten, molybdenum, and carbon.

XRD Data is stores in excel file, in separate sheets. The pandas library allows us to import data from these sheets easily.

In [444…

```
# Import data
p1 = pd.read_excel('xrd_data.xlsx', sheet_name='p1', header=None)
p1.columns = ["2theta", "intensity"]

p2 = pd.read_excel('xrd_data.xlsx', sheet_name='p2', header=None)
p2.columns = ["2theta", "intensity"]

p3 = pd.read_excel('xrd_data.xlsx', sheet_name='p3', header=None)
p3.columns = ["2theta", "intensity"]

p1.describe()
```

Out[444…

	2theta	intensity
count	5251.000000	5251.000000
mean	57.500000	23.642735
std	30.319549	33.934569
min	5.000000	0.000000
25%	31.250000	11.000000
50%	57.500000	17.000000
75%	83.750000	24.000000
max	110.000000	465.000000

In [445…

```
# Setup code #
# ############

# Turn off interactive matplotlib output showing
plt.ioff()

# Set theme for plots
plt.style.use('grayscale')
plt.rc('font',  family='serif')
plt.rc('xtick', labelsize='small')
plt.rc('ytick', labelsize='small')
#plt.rcParams['text.usetex'] = True


# Set image size and resolution
fig, ax = plt.subplots(
    figsize=(8,6),      # 8 inches in width, 6 inch in height
    dpi=600);           # Higher dpi for print quality pictures. Go higher if possible.
```

In [446…

```
# # Plot lines #
# ##############

ax.plot(
```

```
    p1['2theta'].to_numpy(),            # x data
    p1['intensity'].to_numpy() + 600*2 # Y data
    #label = 'P1 (Stellite 1)'           # legend label
)

ax.plot(
    p2['2theta'].to_numpy(),               # x data
    p2['intensity'].to_numpy() + 600*1     # Y data
    #label = 'P2 (50:50 blend of P1 and P2)' # legend label
)

ax.plot(
    p3['2theta'].to_numpy(),            # x data
    p3['intensity'].to_numpy() + 600*0 # Y data
    #label = 'P3 (Stellite 21)'          # legend label
);
```

In [447…

```
# # Ploting alpha cobalt with a black square #
# ##########################################

# Yeah, I just moved stuff around till it looked good. - Vi

ax.axvline(x = 91, color = 'grey', alpha = 0.4, linewidth = '0.5')
ax.plot(91,115+600*2, marker='s', color='indianred')
ax.plot(91,155+600*1, marker='s', color='indianred')
ax.plot(91,155+600*0, marker='s', color='indianred')

ax.axvline(x = 75, color = 'grey', alpha = 0.4, linewidth = '0.5')
ax.plot(75,130+600*2, marker='s', color='indianred')
ax.plot(75,170+600*1, marker='s', color='indianred')
ax.plot(75,170+600*0, marker='s', color='indianred')

ax.axvline(x = 51, color = 'grey', alpha = 0.4, linewidth = '0.5')
ax.plot(51,150+600*2, marker='s', color='indianred')
ax.plot(51,170+600*1, marker='s', color='indianred')
ax.plot(51,200+600*0, marker='s', color='indianred')

ax.axvline(x = 43.8, color = 'grey', alpha = 0.4, linewidth = '0.5')
ax.plot(43.8,510+600*2, marker='s', color='indianred')
ax.plot(43.8,480+600*1, marker='s', color='indianred')
```

```
ax.plot(43.8,470+600*0, marker='s', color='indianred',
        label=r'$\alpha-Co$', linestyle='dashed', linewidth = '0.1')
```

Out[447… `[<matplotlib.lines.Line2D at 0x14851e5d0>]`

In [448…
```
# # Axis options #
# ################

# Set axis labels and legends
ax.set_xlabel(r"$2\theta$ $(deg)$", fontsize=12)
ax.set_ylabel('Intensity (au)', fontsize=12)

# Set axis limits
ax.set_xlim(left=30, right=100)
ax.set_ylim(bottom=-100, top=600*3+200)

# Hide y axis ticks and numbers
ax.yaxis.set_tick_params(labelleft=False)
ax.set_yticks([])

# Set x axis ticks
from matplotlib.ticker import (MultipleLocator, AutoMinorLocator)
ax.xaxis.set_major_locator(MultipleLocator(10))
ax.xaxis.set_major_formatter('{x:.0f}')
# For the minor ticks, use no labels; default NullFormatter.
ax.xaxis.set_minor_locator(MultipleLocator(5))
```

In [449…
```
# # Typing names of each XRD. Don't want the lines in the legend #
# ################################################################

ax.text(31, 55 + 600*2, 'P1', fontsize=12);
ax.text(31, 55 + 600*1, 'P2', fontsize=12);
ax.text(31, 55 + 600*0, 'P3', fontsize=12);
```

In [450…
```
# # Plotting Cr23C3 and Cr7C3 with stars #
# ########################################

# Plot Cr23C3
#ax.axvline(x = 90, color = 'grey', alpha = 0.4, linewidth = '0.5')
#ax.plot(90,50+600*0, marker='*', color='k')
```

```
#ax.plot(90,480+600*1, marker='*', color='k')
ax.plot(90,75+600*0, marker='X', color='steelblue',
        label=r'$Cr_{23}C_3$', linestyle='dashed', linewidth = '0.1')


# Plot Cr7C3
ax.axvline(x = 38.8, color = 'grey', alpha = 0.4, linewidth = '0.5')
ax.axvline(x = 42.7, color = 'grey', alpha = 0.4, linewidth = '0.5')
ax.axvline(x = 50, color = 'grey', alpha = 0.4, linewidth = '0.5')
ax.plot(42.7,280+600*2, marker='*', color='steelblue')
ax.plot(38.8,80+600*1, marker='*', color='steelblue')
ax.plot(50,80+600*2, marker='*', color='steelblue')
ax.plot(50,80+600*1, marker='*', color='steelblue')
ax.plot(38.8,90+600*2, marker='*', color='steelblue',
        label=r'$Cr_{7}C_3$', linestyle='dashed', linewidth = '0.1')
```

Out[450… [<matplotlib.lines.Line2D at 0x1483d52d0>]

In [451…

```
# # Plotting W compounds #
# ########################

# Co3W
ax.plot(61.5,165+600*1, marker='P', color='k', markerfacecolor='none')
ax.plot(61.5,105+600*2, marker='P', color='k', markerfacecolor='none')
ax.plot(62.5,105+600*2, marker='P', color='k', markerfacecolor='none')
ax.plot(73.5,155+600*1, marker='P', color='k', markerfacecolor='none')
ax.plot(46.8,310+600*1, marker='P', color='k', markerfacecolor='none')
ax.plot(46.8,330+600*2, marker='P', color='k', markerfacecolor='none',
        label=r'$Co_3W$', linestyle='dashed', linewidth = '0.1');

# Co3W3C
#ax.axvline(x = 35.6, color = 'grey', alpha = 0.4, linewidth = '0.5')
ax.plot(35.6,105+600*2, marker='<', color='k', markerfacecolor='none')
#ax.axvline(x = 40, color = 'grey', alpha = 0.4, linewidth = '0.5')
ax.plot(40,170+600*2, marker='<', color='k', markerfacecolor='none')
ax.plot(46.8,380+600*2, marker='<', color='k', markerfacecolor='none',
        label=r'$Co_3W_3C$', linestyle='dashed', linewidth = '0.1');


# Co6W6C
```

```
ax.plot(42.7,145+600*1, marker='D', color='k', markerfacecolor='none');
ax.plot(42.7,330+600*2, marker='D', color='k', markerfacecolor='none');
ax.plot(46.8,360+600*1, marker='D', color='k', markerfacecolor='none');
ax.plot(46.8,360+600*1, marker='D', color='k', markerfacecolor='none');
ax.plot(59.7,70+600*2, marker='D', color='k', markerfacecolor='none');
ax.plot(73.5,105+600*2, marker='D', color='k', markerfacecolor='none',
        label=r'$Co_6W_6C$', linestyle='dashed', linewidth = '0.1');
```

In [452…

```
# # Plotting Mo compounds #
# ########################


# Co3Mo
ax.axvline(x = 41, color = 'grey', alpha = 0.4, linewidth = '0.5')
ax.plot(41,110+600*0, marker='P', color='k')
ax.plot(41,100+600*1, marker='P', color='k')
ax.axvline(x = 61.5, color = 'grey', alpha = 0.4, linewidth = '0.5')
ax.plot(61.5,100+600*1, marker='P', color='k')
ax.plot(61.5,100+600*0, marker='P', color='k')
#ax.axvline(x = 73.5, color = 'grey', alpha = 0.4, linewidth = '0.5')
ax.plot(73.5,100+600*1, marker='P', color='k')
#ax.axvline(x = 83.5, color = 'grey', alpha = 0.4, linewidth = '0.5')
ax.plot(83.5,150+600*0, marker='P', color='k',
                label=r'$Co_3Mo$', linestyle='dashed', linewidth = '0.1');
ax.axvline(x = 46.8, color = 'grey', alpha = 0.4, linewidth = '0.5')
ax.plot(46.8,380+600*0, marker='P', color='k')

# Co2Mo3
ax.axvline(x = 40.6, color = 'grey', alpha = 0.4, linewidth = '0.5')
ax.plot(40.6,175+600*1, marker='<', color='k')
ax.plot(40.6,175+600*0, marker='<', color='k')
ax.plot(46.8,430+600*0, marker='<', color='k',
        label=r'$Co_2Mo_3$', linestyle='dashed', linewidth = '0.1');
ax.plot(42.7,160+600*0, marker='<', color='k')
ax.plot(46.8,260+600*1, marker='<', color='k')


# MoC
ax.axvline(x = 42.7, color = 'grey', alpha = 0.4, linewidth = '0.5')
#ax.axvline(x = 83.5, color = 'grey', alpha = 0.4, linewidth = '0.5')
```

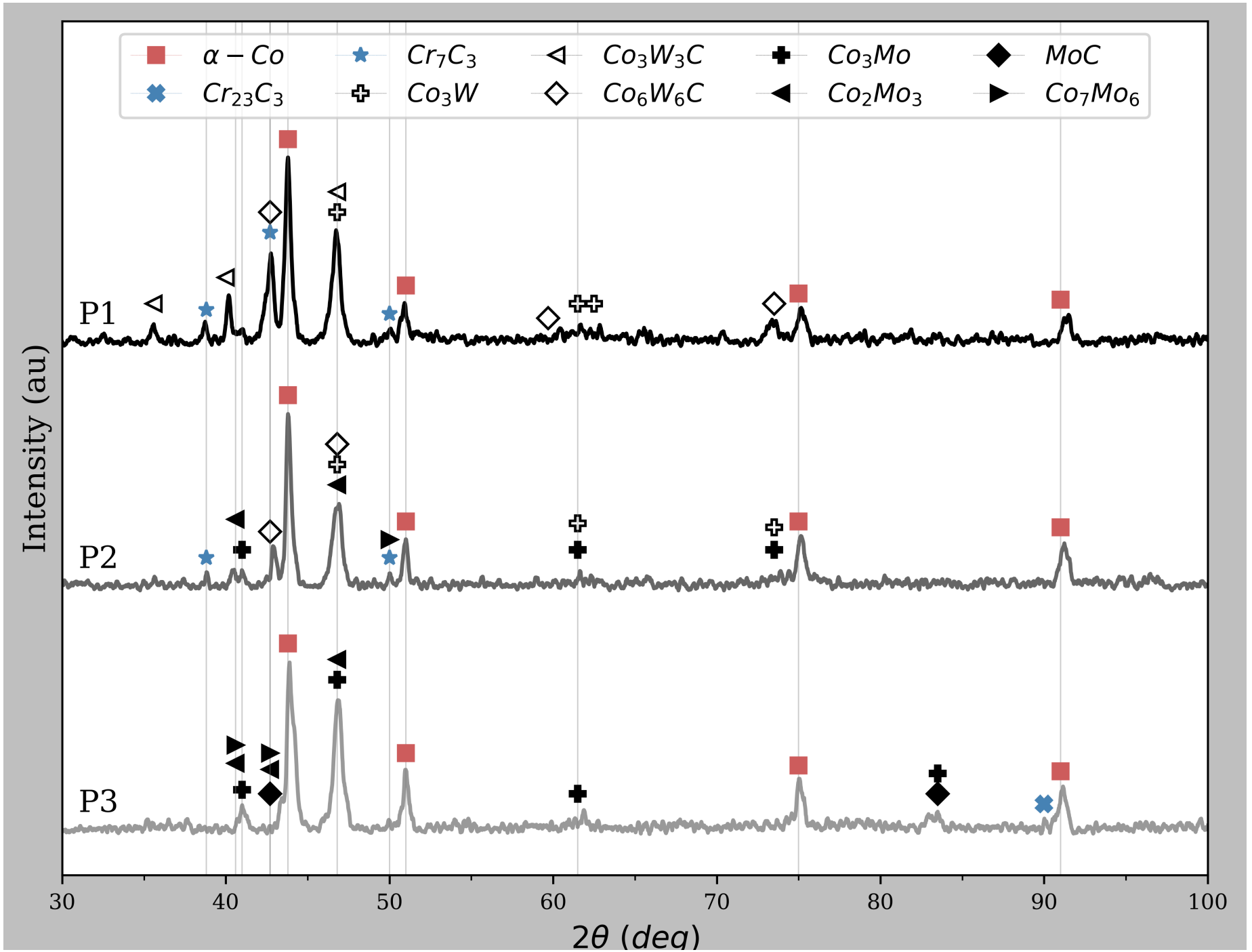
```
ax.plot(42.7,100+600*0, marker='D', color='k')
ax.plot(83.5,100+600*0, marker='D', color='k',
        label=r'$MoC$', linestyle='dashed', linewidth = '0.1');

# Co7Mo6
ax.plot(40.6,220+600*0, marker='>', color='k')
ax.plot(42.7,200+600*0, marker='>', color='k')
ax.plot(50,125+600*1, marker='>', color='k',
        label=r'$Co_7Mo_6$', linestyle='dashed', linewidth = '0.1');
```

In [453…

```
# # Legend options #
# ##################

# Show legend for plot?
# ax.legend() # .set_visible(False)
ax.legend(loc="upper center", ncol=5);
```

In [454…

```
# # Show image and save to appropriate file name #
# ################################################

plt.show()
fig.savefig(
    'stellite_xrd_comparison.jpeg',        # Name of the generated file
    bbox_inches='tight',  # removes extra white space around figure
    transparent=True      # Transparent background
)
```


α − Co
Cr7C3
Co3W3C
Co3Mo
MoC
Cr23C3
Co3W
Co6W6C
Co2Mo3
Co7Mo6
P1
P2
P3
Intensity (au)
30
40
50
60
70
80
90
100
2θ (deg)

In [ ]:

Bar Plot

	Pin	Pin_error	Disc	Disc_Error	BoF	BoF_Error
D1(HS1)	0.0062	0.0029	0.0120	0.0018	0.0671	0.0143
D3	0.0154	0.0027	0.0275	0.0062	0.3588	0.0155
D5(HS21)	0.0184	0.0084	0.0423	0.0112	0.3638	0.0424

In [455…
```
# # Library import #
# ##################
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
```

In [456…
```
# # Data copied from excel #
# ############################
df = pd.DataFrame(columns=['Pin', 'Pin_Error', 'Disc', 'Disc_Error', 'BoF', 'BoF_Error'])
df.loc[0] = [0.0062, 0.0029, 0.0120, 0.0018, 0.0671, 0.0143]
df.loc[1] = [0.0154, 0.0027, 0.0275, 0.0062, 0.3588, 0.0155]
df.loc[2] = [0.0184, 0.0084, 0.0423, 0.0112, 0.3638, 0.0424]
df = df.set_index(pd.Index(["P1", "P2", "P3"]))
```

In [457…
```
# # Set theme for plots #
# ########################

plt.style.use('grayscale')
plt.rc('font',  family='serif')
plt.rc('xtick', labelsize='small')
plt.rc('ytick', labelsize='small')
#plt.rcParams['text.usetex'] = True
plt.ioff(); # Turn off interactive matplotlib output showing
```

In [458…
```
# # Set image size, resolution, bar widths #
# ##########################################
fig, ax = plt.subplots(
```

```
    figsize=(8,6),      # 8 inches in width, 6 inch in height
    dpi=600);           # Higher dpi for print quality pictures. Go higher if possible.
twin1 = ax.twinx()
twin2 = ax.twinx()
twin2.spines.left.set_position(("axes", -0.1))
twin2.yaxis.set_label_position('left')
twin2.yaxis.set_ticks_position('left')

barWidth = 0.45 # set width of bar
bar1 = np.arange(len(df["Pin"].to_numpy()))  # For experimental x positions
bar2 = 3.5 + np.arange(len(df["Pin"].to_numpy()))  # For normalized x positions
```

In [459…

```
# # Pin and Disk Bar Plot #
# ########################


# Disc Bar Plot
twin2.bar(bar1, df["Disc"].to_numpy(),
       width = barWidth, label="PoD-Disc",
       color='lightblue', hatch='/',
       edgecolor = "black", linewidth = 1,
       bottom=np.zeros(3))
for x, value, err in zip(bar1, df["Disc"].to_numpy(), df["Disc_Error"].to_numpy()):
    twin2.errorbar(x+barWidth/6, value,
                yerr=err,
                color='black',
                capsize=5, elinewidth=1, markeredgewidth=1)
for x, offset, value, err, offset_err in zip(bar1, df["Pin"].to_numpy(),
                                             df["Disc"].to_numpy(),
                                             df["Disc_Error"].to_numpy(),
                                             df["Pin_Error"].to_numpy()):
    twin2.text(
        x+barWidth/6, (offset+value+offset_err)+0.001,
        f"${value:.3f} \pm {err:.3f}$",
        ha = 'center',
        rotation='vertical',
        fontsize = 'x-small'
    )
```

```
# Pin Bar Plot
twin2.bar(bar1, df["Pin"].to_numpy(),
       width = barWidth, label="PoD-Pin",
       color='lightcoral', #hatch="\\\\",
       edgecolor = "black", linewidth = 1,
       bottom=df["Disc"].to_numpy())
for x, value, err in zip(bar1, df["Disc"].to_numpy()+df["Pin"].to_numpy(), df["Pin_Error"].to_numpy()):
    twin2.errorbar(x-barWidth/6, value,
                yerr=err,
                color='black',
                capsize=5, elinewidth=1, markeredgewidth=1)
    twin2.text(
        x-barWidth/6, (value+err)+0.001,
        f"${value:.3f} \pm {err:.3f}$",
        ha = 'center',
        rotation='vertical',
        fontsize = 'x-small'
    )
```

In [460…

```
# # BoF Bar Plot #
# ################
ax.bar(bar1 + barWidth, df["BoF"].to_numpy(),
       width = barWidth, label="BoF",
       edgecolor = "black", linewidth = 1,
       color='white', hatch='...',
       bottom=np.zeros(3))
for x, value, err in zip(bar1 + barWidth, df["BoF"].to_numpy(), df["BoF_Error"].to_numpy()):
    ax.errorbar(x, value,
                yerr=err,
                color='black',
                capsize=5, elinewidth=1, markeredgewidth=1)
    ax.text(
        x, (value+err)+0.005,
        f"${value:.3f} \pm {err:.3f}$",
        ha = 'center',
        rotation='vertical',
        fontsize = 'x-small'
    )
```

In [461…

```
# # Pin and Disk Normalized Bar Plot #
```

```
# ###################################

# Disc Bar Plot
twin1.bar(
    bar2, df["Disc"].to_numpy()/(25*100),
    width = barWidth, label="PoD-Disc",
    color='lightblue', hatch='/',
    edgecolor = "black", linewidth = 1,
    bottom=np.zeros(3)
)
for x, value, err in zip(bar2, df["Disc"].to_numpy(), df["Disc_Error"].to_numpy()):
    twin1.errorbar(
        x+barWidth/8, value/(25*100),
        yerr=err/(25*100),
        color='black',
        capsize=5, elinewidth=1, markeredgewidth=1
    )

# Pin Normalized Bar Plot
twin1.bar(
    bar2, df["Pin"].to_numpy()/(25*100),
    width = barWidth, label="PoD-Pin",
    color='lightcoral', #hatch="\\\\",
    edgecolor = "black", linewidth = 1,
    bottom=df["Disc"].to_numpy()/(25*100)
)
for x, value, err in zip(bar2, df["Disc"].to_numpy()+df["Pin"].to_numpy(), df["Pin_Error"].to_numpy()):
    twin1.errorbar(
        x-barWidth/6, value/(25*100),
        yerr=err/(25*100),
        color='black',
        capsize=5, elinewidth=1, markeredgewidth=1
    )
    twin1.text(
        x-barWidth/6, (value+err)/(25*100)+0.05e-5,
        f"${1e5*value/(25*100):.2f} \pm {1e5*err/(25*100):.2f}$",
        ha = 'center',
        rotation='vertical',
        fontsize = 'x-small'
    )
```

```
for x, offset, value, err, offset_err in zip(bar2, df["Pin"].to_numpy(),
                                             df["Disc"].to_numpy(),
                                             df["Disc_Error"].to_numpy(),
                                             df["Pin_Error"].to_numpy()):
    twin1.text(
        x+barWidth/6, (offset+value+offset_err)/(25*100)+0.05e-5,
        f"${1e5*value/(25*100):.2f} \pm {1e5*err/(25*100):.2f}$",
        ha = 'center',
        rotation='vertical',
        fontsize = 'x-small'
    )
```

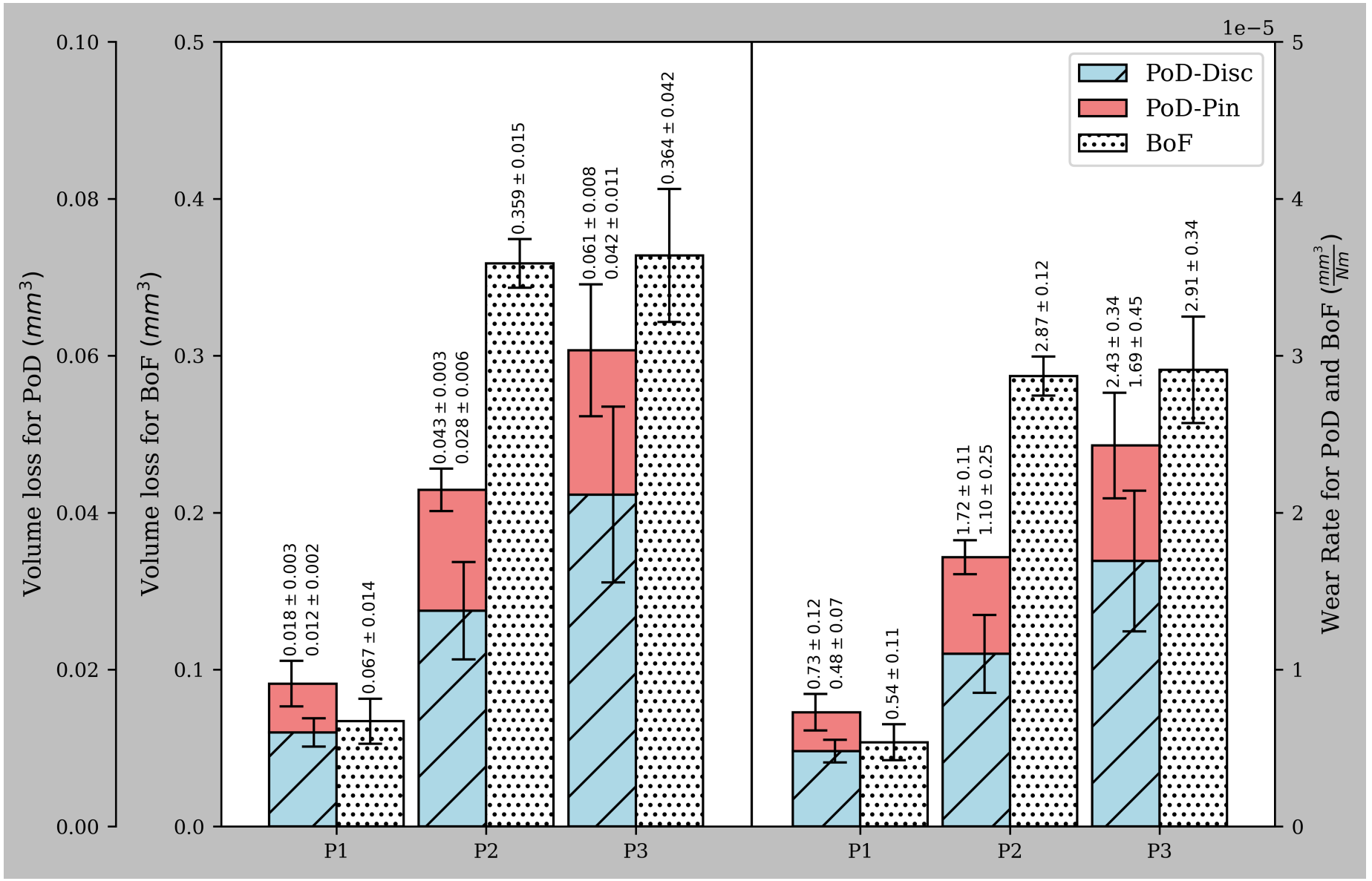
In [462…

```
# # BoF Normalized Bar Plot #
# ##########################

twin1.bar(
    bar2 + barWidth, df["BoF"].to_numpy()/(25*500),
    width = barWidth, label="BoF",
    edgecolor = "black", linewidth = 1,
    color='white', hatch='...',
    bottom=np.zeros(3))
for x, value, err in zip(bar2 + barWidth, df["BoF"].to_numpy(), df["BoF_Error"].to_numpy()):
    twin1.errorbar(
        x, value/(25*500),
        yerr=err/(25*500),
        color='black',
        capsize=5, elinewidth=1, markeredgewidth=1
    )
    twin1.text(
        x, (value+err)/(25*500)+0.05e-5,
        f"${1e5*value/(25*500):.2f} \pm {1e5*err/(25*500):.2f}$",
        ha = 'center',
        rotation='vertical',
        fontsize = 'x-small'
    )
```

In [463…

```
# # Setting axis labels and limits #
# ##################################

ax.axvline(x = 3, color = 'black', alpha = 1, linewidth = '1')
```

```
twin1.set_ylim(top=5e-5)
ax.set_ylim(top=0.5)
twin2.set_ylim(top=0.1)

# use the plt.xticks function to custom labels
ax.set_xticks(
    np.append(bar1+barWidth/2, bar2+barWidth/2), # x axis stack
    ("P1","P2","P3",
    "P1","P2","P3")
)

twin2.set_ylabel("Volume loss for PoD $(mm^3)$")
ax.set_ylabel("Volume loss for BoF $(mm^3)$")
twin1.set_ylabel(r"Wear Rate for PoD and BoF $(\frac{mm^3}{Nm})$")

#twin1.set_ylabel("Temperature")


#ax.set_title("Number of penguins with above average body mass")
twin1.legend(loc="best")
#plt.tight_layout()
plt.show()

fig.savefig(
    'stellite_volume_loss_comparison.jpeg',        # Name of the generated file
    bbox_inches='tight',  # removes extra white space around figure
    transparent=True      # Transparent background
)
```



Sand Wear Plot

In [464…
```
# # Library import #
# ##################
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
```

In [465…
```
# # Data copied from excel #
# ##########################
df = pd.DataFrame(columns=['SandA', 'SandA_Error', 'SandB', 'SandB_Error'])
df.loc[0] = [3.7, 0.12, 3.73, 0.31]
df.loc[1] = [7.37, 0.65, 6.64, 1.53]
df.loc[2] = [18.69, 0.05, 25.54, 0.33]
df = df.set_index(pd.Index(["P1", "P2", "P3"]))

df
```

Out[465…

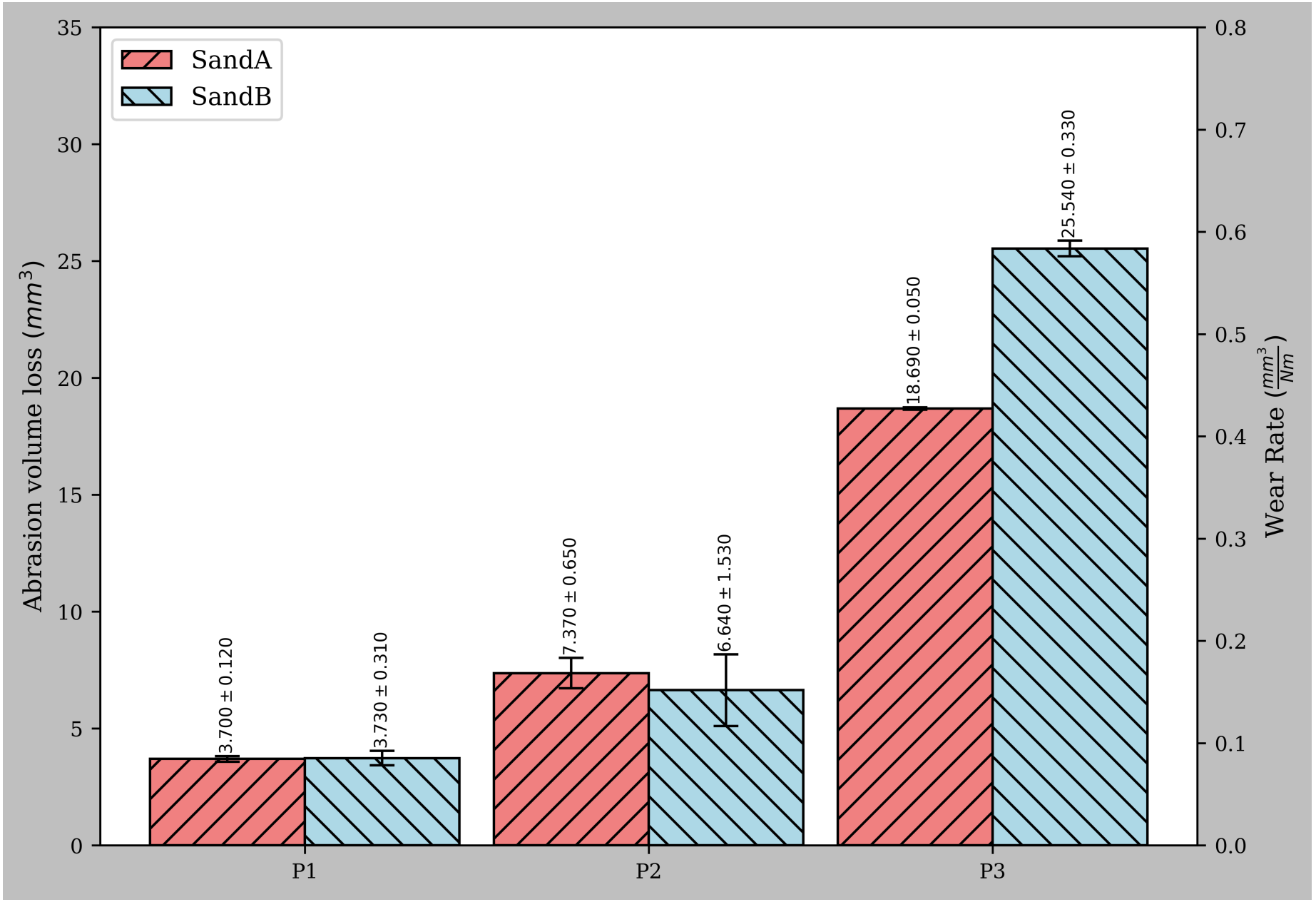
	SandA	SandA_Error	SandB	SandB_Error
P1	3.70	0.12	3.73	0.31
P2	7.37	0.65	6.64	1.53
P3	18.69	0.05	25.54	0.33

In [466…
```
# # Set theme for plots #
# ########################

plt.style.use('grayscale')
plt.rc('font',  family='serif')
plt.rc('xtick', labelsize='small')
plt.rc('ytick', labelsize='small')
#plt.rcParams['text.usetex'] = True
plt.ioff(); # Turn off interactive matplotlib output showing
```

In [467…
```
# # Set image size, resolution, bar widths #
# ##########################################
fig, ax = plt.subplots(
    figsize=(8,6),      # 8 inches in width, 6 inch in height
    dpi=600);           # Higher dpi for print quality pictures. Go higher if possible.
```

```
barWidth = 0.45 # set width of bar
sandA_bar = -barWidth/2 + np.arange(len(df["SandA"].to_numpy()))  # For sandA x positions
sandB_bar =  barWidth/2 + np.arange(len(df["SandB"].to_numpy()))

# # SandA Bar Plot #
# ################
ax.bar(sandA_bar, df["SandA"].to_numpy(),
       width = barWidth, label="SandA",
       edgecolor = "black", linewidth = 1,
       color='lightcoral', hatch='//',
       bottom=np.zeros(3))
for x, value, err in zip(sandA_bar, df["SandA"].to_numpy(), df["SandA_Error"].to_numpy()):
    ax.errorbar(x, value,
                yerr=err,
                color='black',
                capsize=5, elinewidth=1, markeredgewidth=1)
    ax.text(
        x, (value+err)+0.25,
        f"${value:.3f} \pm {err:.3f}$",
        ha = 'center',
        rotation='vertical',
        fontsize = 'x-small'
    )

# # SandA Bar Plot #
# ################
ax.bar(sandB_bar, df["SandB"].to_numpy(),
       width = barWidth, label="SandB",
       edgecolor = "black", linewidth = 1,
       color='lightblue', hatch='\\\\',
       bottom=np.zeros(3))
for x, value, err in zip(sandB_bar, df["SandB"].to_numpy(), df["SandB_Error"].to_numpy()):
    ax.errorbar(x, value,
                yerr=err,
                color='black',
                capsize=5, elinewidth=1, markeredgewidth=1)
    ax.text(
        x, (value+err)+0.25,
        f"${value:.3f} \pm {err:.3f}$",
        ha = 'center',
        rotation='vertical',
```

```
        fontsize = 'x-small'
    )

ax.legend(loc="upper left")
ax.set_ylabel("Abrasion volume loss $(mm^3)$")
ax.set_ylim(top=35)


# use the plt.xticks function to custom labels
ax.set_xticks(
    (sandA_bar+sandB_bar)/2, # x axis stack
    ("P1","P2","P3")
)


ax2 = ax.twinx()  # instantiate a second axes that shares the same x-axis
ax2.set_ylim(top=0.8)
ax2.set_ylabel(r"Wear Rate $( \frac{mm^3}{Nm} )$")

plt.show()

fig.savefig(
    'stellite_sand_abrasion_comparison.jpeg',        # Name of the generated file
    bbox_inches='tight',  # removes extra white space around figure
    transparent=True      # Transparent background
)
```


SandA
SandB
Abrasion volume loss (mm^3)
Wear Rate ($\frac{mm^3}{Nm}$)
3.700 ± 0.120
3.730 ± 0.310
7.370 ± 0.650
6.640 ± 1.530
18.690 ± 0.050
25.540 ± 0.330
P1
P2
P3

In [ ]:

In [ ]: