Load Experiment 03 from old dataset

In [8]:

```
from ae_measure2 import load_PLB
mypath = 'E:/file_cabinet/phd/projects/aeml/data/natfreq/220617_natfreqdataset.json'
data = load_PLB(mypath)

from ae_functions import create_figure, plot_signal
import matplotlib.pyplot as plt
from ae_functions import get_signal_start_end

waves = data['waves']
location = data['location']
angle = data['angle']
length = data['length']
event = data['event']

import numpy as np

w2in_03 = []
w4in_03 = []
w6in_03 = []
w8in_03 = []

for idx,wave in enumerate(waves):

    if length[idx] == '2in' and location[idx] == 'tope':
        w2in_03.append(wave)
    if length[idx] == '4in' and location[idx] == 'tope':
        w4in_03.append(wave)
    if length[idx] == '6in' and location[idx] == 'tope':
        w6in_03.append(wave)
    if length[idx] == '8in' and location[idx] == 'tope':
        w8in_03.append(wave)

print('Experiment 03')
print(f"# of 2 in top plate waves : {len(w2in_03)}")
print(f"# of 4 in top plate waves : {len(w4in_03)}")
print(f"# of 6 in top plate waves : {len(w6in_03)}")
print(f"# of 8 in top plate waves : {len(w8in_03)}\n")
```

```
Experiment 03
# of 2 in top plate waves : 34
# of 4 in top plate waves : 36
# of 6 in top plate waves : 37
# of 8 in top plate waves : 86
```

Load in Experiment 04 - this is when we changed the filtering process, so need to update above code, its just using the older .json dataset

In [9]:

```
mypath = 'E:/file_cabinet/phd/projects/aeml/data/natfreq/experiment04/20220629_experime
data = load_PLB(mypath)
```

```
waves = data['waves']
location = data['location']
angle = data['angle']
length = data['length']
event = data['event']

import numpy as np

# 2 inch
w2in_04 = waves[np.where(length=='2in')]
ev2in_04 = event[np.where(length=='2in')]
print(f"# of 2 in plate waves : {len(w2in_04)}")

# 4 inch
w4in_04 = waves[np.where(length=='4in')]
ev4in_04 = event[np.where(length=='4in')]
print(f"# of 4 in plate waves : {len(w4in_04)}")

# 6 inch
w6in_04 = waves[np.where(length=='6in')]
ev6in_04 = event[np.where(length=='6in')]
print(f"# of 6 in plate waves : {len(w6in_04)}")

# 8 inch
w8in_04 = waves[np.where(length=='8in')]
ev8in_04 = event[np.where(length=='8in')]
print(f"# of 8 in plate waves : {len(w8in_04)}")
```

```
# of 2 in plate waves : 35
# of 4 in plate waves : 46
# of 6 in plate waves : 43
# of 8 in plate waves : 52
```

Plot the waves from each plate length, and ensure no signals need to be filtered

Double signals outliers removed (such as signals that start way earlier).

In [10]:

```
# Signal Processing Parameters
sig_len = 1024
dt = 10**-7
duration = sig_len*dt*10**6 # convert to us
time = np.linspace(0,duration,sig_len) # discretization of signal time
```

Compute and Plot the Mean Waveforms for each Plate Length

In [11]:

```
# 2 in_03
mean_w2in_03 = np.mean(w2in_03, axis=0) # average each column over all examples
std_w2in_03 = np.std(w2in_03, axis=0)   # standard deviation

# 4 in_03
mean_w4in_03 = np.mean(w4in_03, axis=0)
std_w4in_03 = np.std(w4in_03, axis=0)
```

```
# 6 in_03
mean_w6in_03 = np.mean(w6in_03, axis=0)
std_w6in_03 = np.std(w6in_03, axis=0)

# 8 in_03
mean_w8in_03 = np.mean(w8in_03, axis=0)
std_w8in_03 = np.std(w8in_03, axis=0)
```

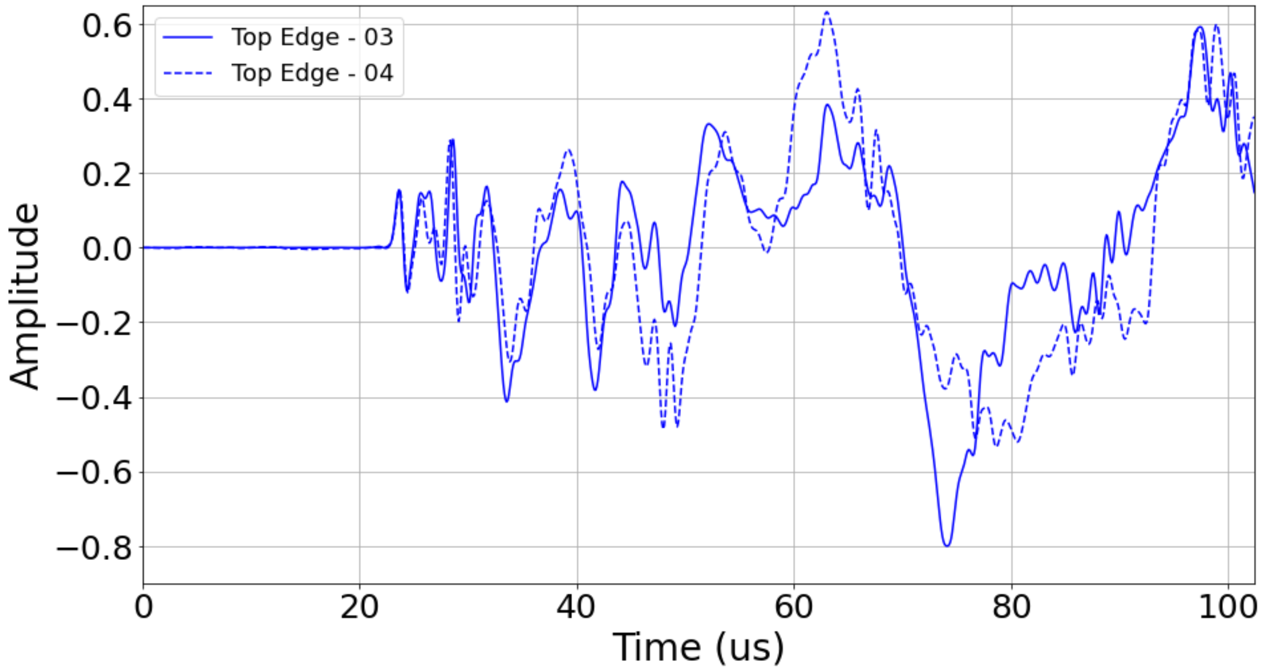
In [12]:

```
# 2 in_04
mean_w2in_04 = np.mean(w2in_04, axis=0) # average each column over all examples
std_w2in_04 = np.std(w2in_04, axis=0)   # standard deviation

# 4 in_04
mean_w4in_04 = np.mean(w4in_04, axis=0)
std_w4in_04 = np.std(w4in_04, axis=0)

# 6 in_04
mean_w6in_04 = np.mean(w6in_04, axis=0)
std_w6in_04 = np.std(w6in_04, axis=0)

# 8 in_04
mean_w8in_04 = np.mean(w8in_04, axis=0)
std_w8in_04 = np.std(w8in_04, axis=0)
```

2 in for all three different orientations

In [18]:

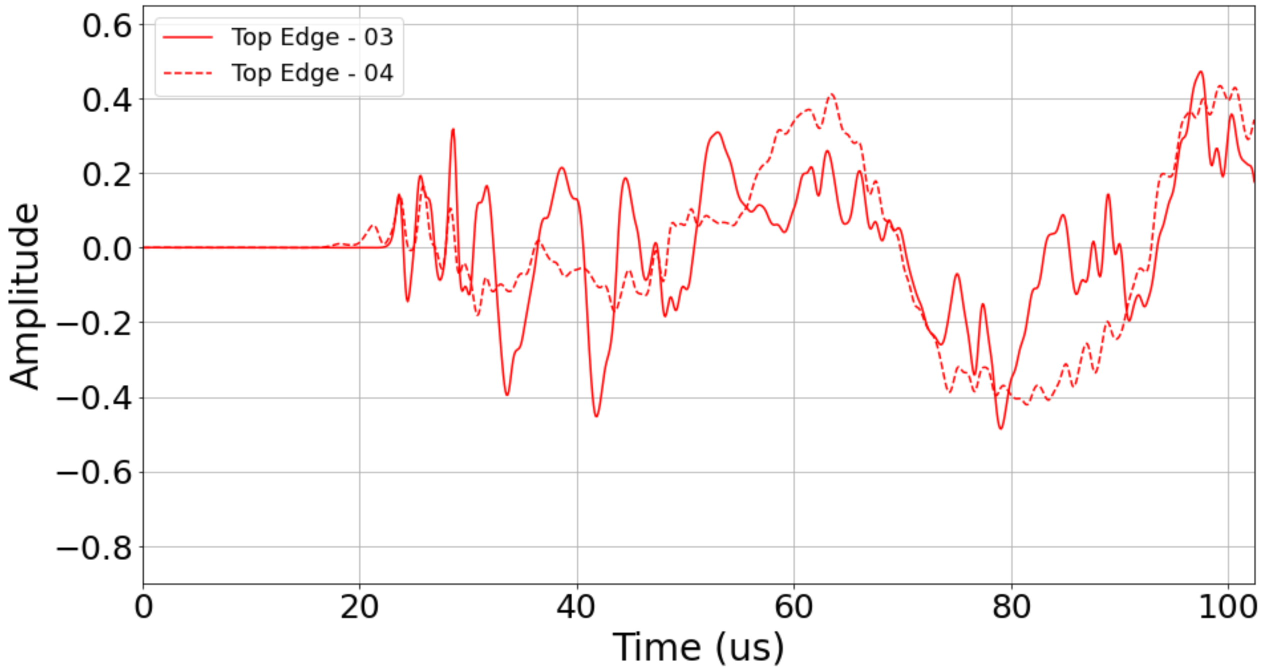
```
# Plot the averaged waveforms
fig,spec2 = create_figure('',columns=1,rows=1,width=15,height=8,default_font_size=18\
                         ,tick_font_size=25,legend_font_size=18,axes_font_size=28,
                         title_font_size=28)
sig_len = 1024
dt = 10**-7
duration = sig_len*dt*10**6 # convert to us
time = np.linspace(0,duration,sig_len) # discretization of signal time

ax = fig.add_subplot(spec2[0,0])
ax.plot(time,mean_w2in_03,'-',label = 'Top Edge - 03', color='blue')
ax.plot(time,mean_w2in_04,'--',label = 'Top Edge - 04', color='blue')
plt.legend()
ax.set_xlim([0,duration])
ax.set_ylim([-0.9,0.65])
ax.set_xlabel('Time (us)')
ax.set_ylabel('Amplitude')

plt.grid()
plt.show()
```



4 in

In [19]:

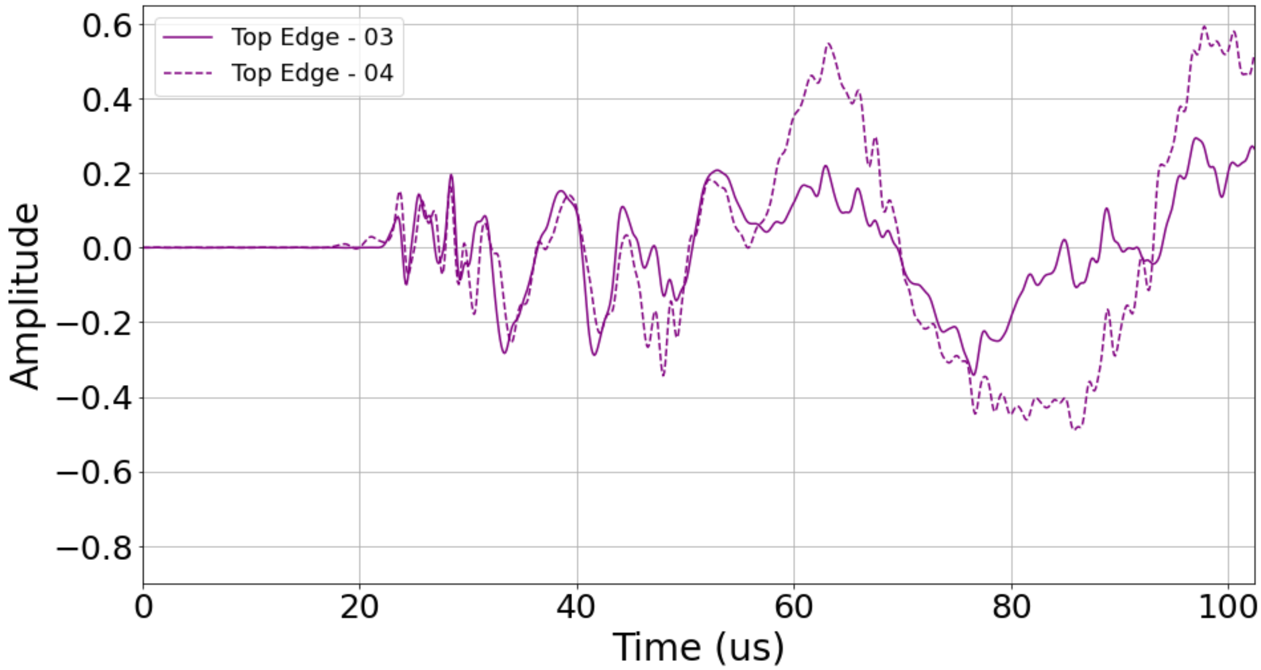
```
# Plot the averaged waveforms
fig,spec2 = create_figure('',columns=1,rows=1,width=15,height=8,default_font_size=18\
                         ,tick_font_size=25,legend_font_size=18,axes_font_size=28,
                         title_font_size=28)
sig_len = 1024
dt = 10**-7
duration = sig_len*dt*10**6 # convert to us
time = np.linspace(0,duration,sig_len) # discretization of signal time

ax = fig.add_subplot(spec2[0,0])
ax.plot(time,mean_w4in_03,'-',label = 'Top Edge - 03', color='red')
ax.plot(time,mean_w4in_04,'--',label = 'Top Edge - 04', color='red')
plt.legend()
ax.set_xlim([0,duration])
ax.set_ylim([-0.9,0.65])
ax.set_xlabel('Time (us)')
ax.set_ylabel('Amplitude')

plt.grid()
plt.show()
```



6 in

In [20]:

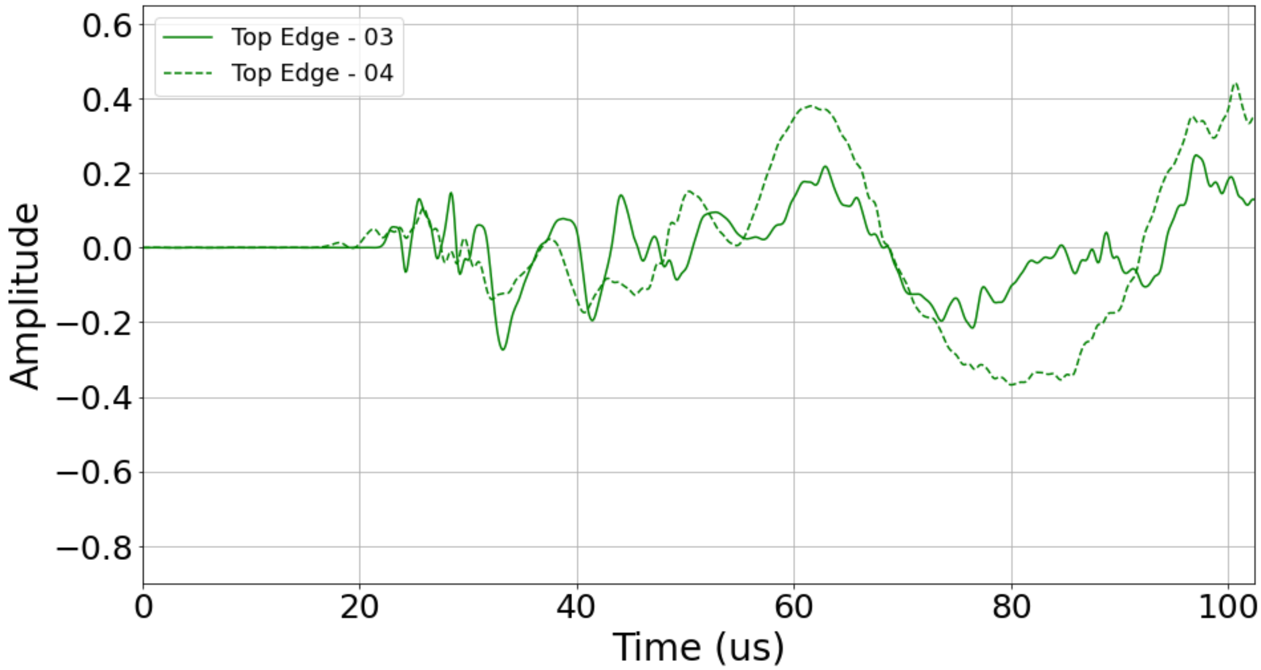
```
# Plot the averaged waveforms
fig,spec2 = create_figure('',columns=1,rows=1,width=15,height=8,default_font_size=18\
                         ,tick_font_size=25,legend_font_size=18,axes_font_size=28,
                         title_font_size=28)
sig_len = 1024
dt = 10**-7
duration = sig_len*dt*10**6 # convert to us
time = np.linspace(0,duration,sig_len) # discretization of signal time

ax = fig.add_subplot(spec2[0,0])
ax.plot(time,mean_w6in_03,'-',label = 'Top Edge - 03', color='purple')
ax.plot(time,mean_w6in_04,'--',label = 'Top Edge - 04', color='purple')
plt.legend()
ax.set_xlim([0,duration])
ax.set_ylim([-0.9,0.65])
ax.set_xlabel('Time (us)')
ax.set_ylabel('Amplitude')

plt.grid()
plt.show()
```



8 in

In [21]:

```
# Plot the averaged waveforms
fig,spec2 = create_figure('',columns=1,rows=1,width=15,height=8,default_font_size=18\
                         ,tick_font_size=25,legend_font_size=18,axes_font_size=28,
                         title_font_size=28)
sig_len = 1024
dt = 10**-7
duration = sig_len*dt*10**6 # convert to us
time = np.linspace(0,duration,sig_len) # discretization of signal time

ax = fig.add_subplot(spec2[0,0])
ax.plot(time,mean_w8in_03,'-',label = 'Top Edge - 03', color='green')
ax.plot(time,mean_w8in_04,'--',label = 'Top Edge - 04', color='green')
plt.legend()
ax.set_xlim([0,duration])
ax.set_ylim([-0.9,0.65])
ax.set_xlabel('Time (us)')
ax.set_ylabel('Amplitude')

plt.grid()
plt.show()
```



Compute and Plot Mean FFT (Entire Waveform)

The FFTs here are computed on the entire raw waveform (this is necessary to average them all and visualize, otherwise the ffts would have different lengths). Later in the code, the waveforms are chopped down to only what's contained between the start and end (what's visualized in the previous plotting of raw waveforms) before computed frequency domain features.

In [22]:

```
# Compute FFTs
low_pass = 0                # [Hz] ; low frequency cutoff
high_pass = 1000*10**3       # [Hz] ; high frequency cutoff\
dt = 10**-7                 # [seconds] ; sample period / time between samples
fft_units = 1000

from ae_measure2 import fft

# 2 in_03
fft2in_03 = []
for idx,wave in enumerate(w2in_03):
    w,z = fft(dt, wave, low_pass, high_pass)
    fft2in_03.append(z)

fft2in_03 = np.array(fft2in_03)
mean_fft2in_03= np.mean(fft2in_03, axis=0) # average each column over all examples
std_fft2in_03 = np.std(fft2in_03, axis=0)   # standard deviation

# 4 in_03
fft4in_03 = []
for idx,wave in enumerate(w4in_03):
    w,z = fft(dt, wave, low_pass, high_pass)
    fft4in_03.append(z)
fft4in_03 = np.array(fft4in_03)
mean_fft4in_03= np.mean(fft4in_03, axis=0) # average each column over all examples
std_fft4in_03 = np.std(fft4in_03, axis=0)   # standard deviation

# 6 in_03
```

```
fft6in_03 = []
for idx,wave in enumerate(w6in_03):
    w,z = fft(dt, wave, low_pass, high_pass)
    fft6in_03.append(z)
fft6in_03 = np.array(fft6in_03)
mean_fft6in_03= np.mean(fft6in_03, axis=0) # average each column over all examples
std_fft6in_03 = np.std(fft6in_03, axis=0)   # standard deviation

# 8 in_03
fft8in_03 = []
for idx,wave in enumerate(w8in_03):
    w,z = fft(dt, wave, low_pass, high_pass)
    fft8in_03.append(z)
fft8in_03 = np.array(fft8in_03)
mean_fft8in_03= np.mean(fft8in_03, axis=0) # average each column over all examples
std_fft8in_03 = np.std(fft8in_03, axis=0)   # standard deviation

w = w/fft_units; # khz
```

In [23]:

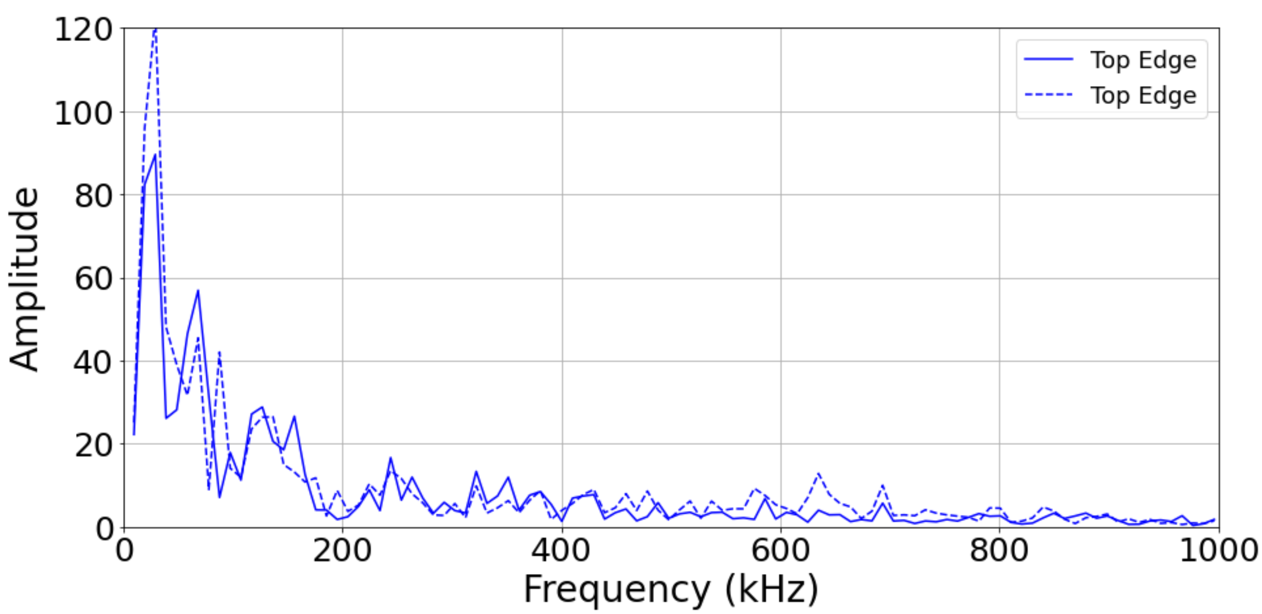
```
# Compute FFTs
low_pass = 0              # [Hz] ; low frequency cutoff
high_pass = 1000*10**3     # [Hz] ; high frequency cutoff\
dt = 10**-7               # [seconds] ; sample period / time between samples
fft_units = 1000

from ae_measure2 import fft

# 2 in_04
fft2in_04 = []
for idx,wave in enumerate(w2in_04):
    w,z = fft(dt, wave, low_pass, high_pass)
    fft2in_04.append(z)

fft2in_04 = np.array(fft2in_04)
mean_fft2in_04= np.mean(fft2in_04, axis=0) # average each column over all examples
std_fft2in_04 = np.std(fft2in_04, axis=0)   # standard deviation

# 4 in_04
fft4in_04 = []
for idx,wave in enumerate(w4in_04):
    w,z = fft(dt, wave, low_pass, high_pass)
    fft4in_04.append(z)
fft4in_04 = np.array(fft4in_04)
mean_fft4in_04= np.mean(fft4in_04, axis=0) # average each column over all examples
std_fft4in_04 = np.std(fft4in_04, axis=0)   # standard deviation

# 6 in_04
fft6in_04 = []
for idx,wave in enumerate(w6in_04):
    w,z = fft(dt, wave, low_pass, high_pass)
    fft6in_04.append(z)
fft6in_04 = np.array(fft6in_04)
mean_fft6in_04= np.mean(fft6in_04, axis=0) # average each column over all examples
std_fft6in_04 = np.std(fft6in_04, axis=0)   # standard deviation

# 8 in_04
fft8in_04 = []
for idx,wave in enumerate(w8in_04):
```

```
    w,z = fft(dt, wave, low_pass, high_pass)
    fft8in_04.append(z)
fft8in_04 = np.array(fft8in_04)
mean_fft8in_04= np.mean(fft8in_04, axis=0) # average each column over all examples
std_fft8in_04 = np.std(fft8in_04, axis=0)   # standard deviation

w = w/fft_units; # khz
```

Plot FFTs for 2 in for all three different orientations

In [25]:

```
# Plot the averaged ffts
fig,spec2 = create_figure('',columns=1,rows=1,width=15,height=7,default_font_size=18\
                         ,tick_font_size=25,legend_font_size=18,axes_font_size=28,
                         title_font_size=28)
ax = fig.add_subplot(spec2[0,0])

ax.plot(w,mean_fft2in_03,'-',label = 'Top Edge', color='blue')
ax.plot(w,mean_fft2in_04,'--',label = 'Top Edge', color='blue')

ax.set_xlim([low_pass/fft_units,high_pass/fft_units])
ax.set_xlabel('Frequency (kHz)')
ax.set_ylabel('Amplitude')
plt.legend()
ax.set_ylim([0,120])

plt.grid()
plt.show()
```

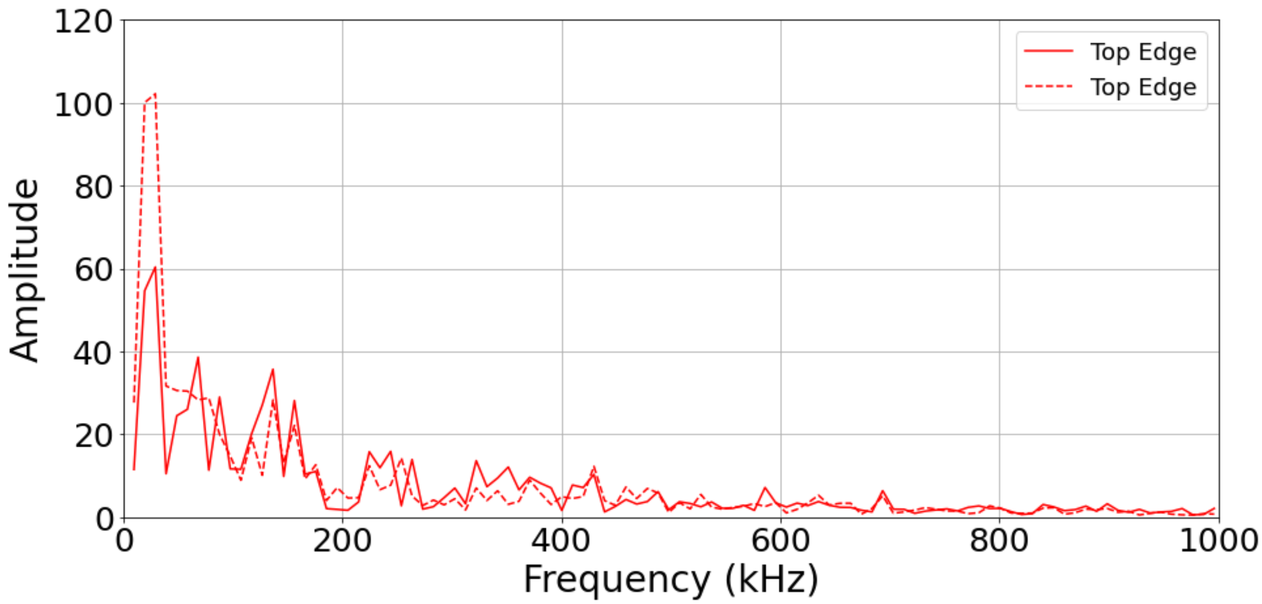


4 in

In [26]:

```
# Plot the averaged ffts
fig,spec2 = create_figure('',columns=1,rows=1,width=15,height=7,default_font_size=18\
                         ,tick_font_size=25,legend_font_size=18,axes_font_size=28,
                         title_font_size=28)
ax = fig.add_subplot(spec2[0,0])

ax.plot(w,mean_fft4in_03,'-',label = 'Top Edge', color='red')
```

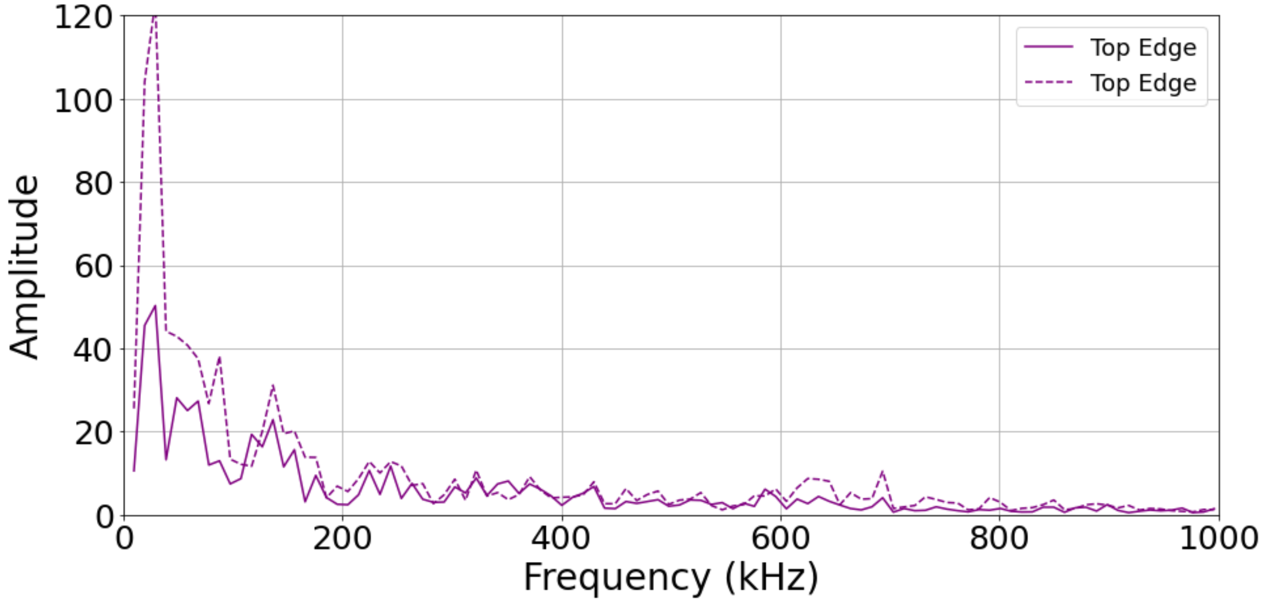
```
ax.plot(w,mean_fft4in_04,'--',label = 'Top Edge', color='red')

ax.set_xlim([low_pass/fft_units,high_pass/fft_units])
ax.set_xlabel('Frequency (kHz)')
ax.set_ylabel('Amplitude')
plt.legend()
ax.set_ylim([0,120])

plt.grid()
plt.show()
```



6 in

In [27]:

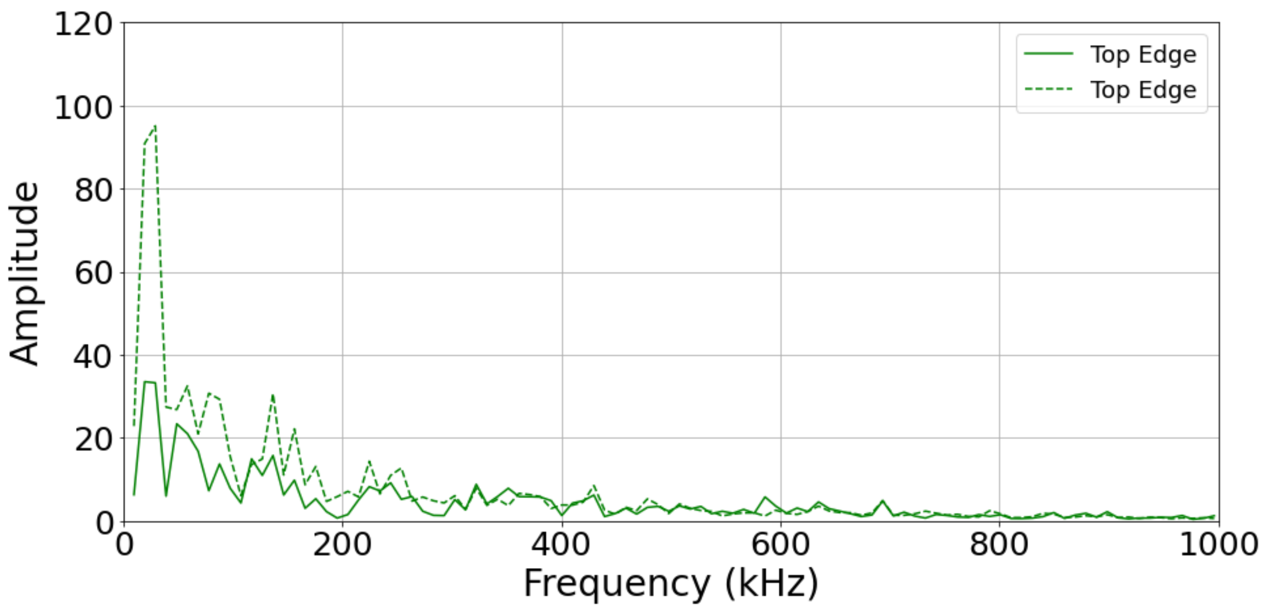
```
# Plot the averaged ffts
fig,spec2 = create_figure('',columns=1,rows=1,width=15,height=7,default_font_size=18\
                          ,tick_font_size=25,legend_font_size=18,axes_font_size=28,
                          title_font_size=28)
ax = fig.add_subplot(spec2[0,0])

ax.plot(w,mean_fft6in_03,'-',label = 'Top Edge', color='purple')
ax.plot(w,mean_fft6in_04,'--',label = 'Top Edge', color='purple')

ax.set_xlim([low_pass/fft_units,high_pass/fft_units])
ax.set_xlabel('Frequency (kHz)')
ax.set_ylabel('Amplitude')
plt.legend()
ax.set_ylim([0,120])

plt.grid()
plt.show()
```



8 in

In [28]:

```
# Plot the averaged ffts
fig,spec2 = create_figure('',columns=1,rows=1,width=15,height=7,default_font_size=18\
                         ,tick_font_size=25,legend_font_size=18,axes_font_size=28,
                         title_font_size=28)
ax = fig.add_subplot(spec2[0,0])

ax.plot(w,mean_fft8in_03,'-',label = 'Top Edge', color='green')
ax.plot(w,mean_fft8in_04,'--',label = 'Top Edge', color='green')

ax.set_xlim([low_pass/fft_units,high_pass/fft_units])
ax.set_xlabel('Frequency (kHz)')
ax.set_ylabel('Amplitude')
plt.legend()
ax.set_ylim([0,120])

plt.grid()
plt.show()
```



Extract Signal Features using ML Framework Code

Get Sause Vector

Note that in feature extraction, the waveform is chopped down to only contain what is in the start and end. The previously plotted ffts were based off the entire waveform, whereas the features extracted here are derivd from the waveform in between start and end markers.

In [29]:

```
from feature_extraction import extract_Sause_vect
low_pass = 0
high_pass = 1000*10**3
# From code docs
# feature_vector = [average_freq, reverb_freq, rise_freq, peak_freq,
#                   freq_centroid, wpf, pp1, pp2, pp3, pp4, pp5, pp6]

sv2in_03 = [] # sause vector
sv4in_03 = [] # sause vector
sv6in_03 = [] # sause vector
sv8in_03 = [] # sause vector

for idx, signal in enumerate(w2in_03):
    sv2in_03.append(extract_Sause_vect(signal,low_pass=low_pass,high_pass=high_pass))

for idx, signal in enumerate(w4in_03):
    sv4in_03.append(extract_Sause_vect(signal,low_pass=low_pass,high_pass=high_pass))

for idx, signal in enumerate(w6in_03):
    sv6in_03.append(extract_Sause_vect(signal,low_pass=low_pass,high_pass=high_pass))

for idx, signal in enumerate(w8in_03):
    sv8in_03.append(extract_Sause_vect(signal,low_pass=low_pass,high_pass=high_pass))

# 2 in_03
mean_sv2in_03 = np.mean(sv2in_03, axis=0) # average each column over all examples
std_sv2in_03 = np.std(sv2in_03, axis=0)   # standard deviation

# 4 in_03
mean_sv4in_03 = np.mean(sv4in_03, axis=0)
std_sv4in_03 = np.std(sv4in_03, axis=0)

# 6 in_03
mean_sv6in_03 = np.mean(sv6in_03, axis=0)
std_sv6in_03 = np.std(sv6in_03, axis=0)

# 8 in_03
mean_sv8in_03 = np.mean(sv8in_03, axis=0)
std_sv8in_03 = np.std(sv8in_03, axis=0)
```

In [30]:

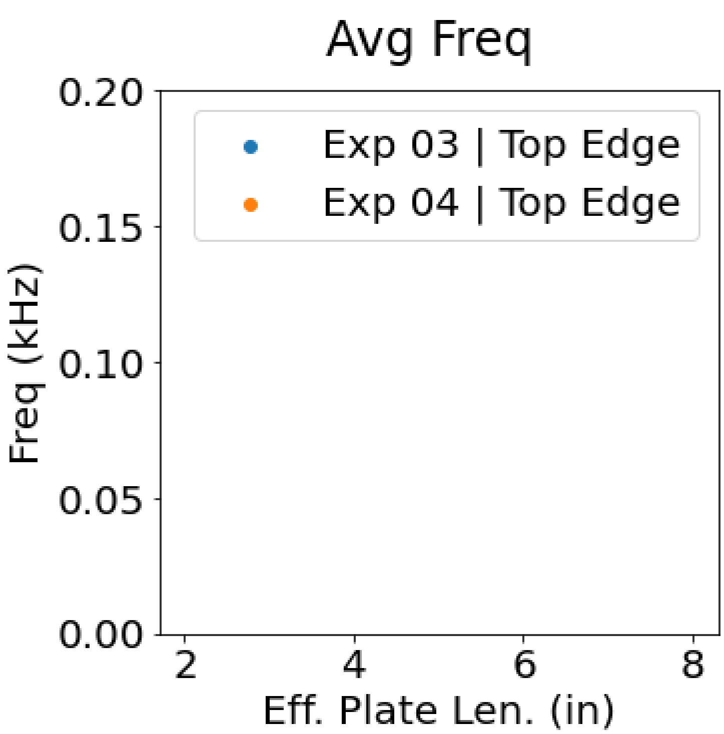
```
from feature_extraction import extract_Sause_vect
low_pass = 0
high_pass = 1000*10**3
# From code docs
```

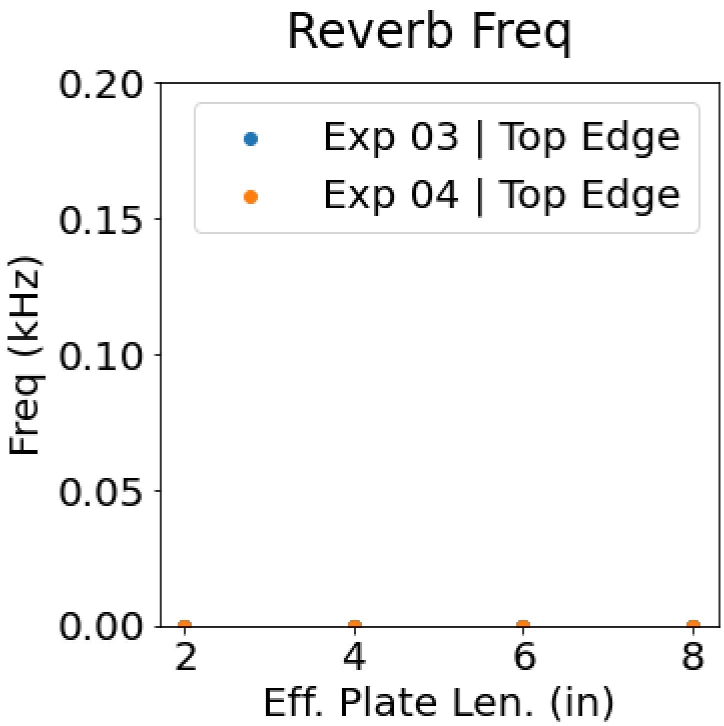
```
# feature_vector = [average_freq, reverb_freq, rise_freq, peak_freq,
#                   freq_centroid, wpf, pp1, pp2, pp3, pp4, pp5, pp6]

sv2in_04 = [] # sause vector
sv4in_04 = [] # sause vector
sv6in_04 = [] # sause vector
sv8in_04 = [] # sause vector

for idx, signal in enumerate(w2in_04):
    sv2in_04.append(extract_Sause_vect(signal,low_pass=low_pass,high_pass=high_pass))

for idx, signal in enumerate(w4in_04):
    sv4in_04.append(extract_Sause_vect(signal,low_pass=low_pass,high_pass=high_pass))

for idx, signal in enumerate(w6in_04):
    sv6in_04.append(extract_Sause_vect(signal,low_pass=low_pass,high_pass=high_pass))

for idx, signal in enumerate(w8in_04):
    sv8in_04.append(extract_Sause_vect(signal,low_pass=low_pass,high_pass=high_pass))

# 2 in_04
mean_sv2in_04 = np.mean(sv2in_04, axis=0) # average each column over all examples
std_sv2in_04 = np.std(sv2in_04, axis=0)   # standard deviation

# 4 in_04
mean_sv4in_04 = np.mean(sv4in_04, axis=0)
std_sv4in_04 = np.std(sv4in_04, axis=0)

# 6 in_04
mean_sv6in_04 = np.mean(sv6in_04, axis=0)
std_sv6in_04 = np.std(sv6in_04, axis=0)

# 8 in_04
mean_sv8in_04 = np.mean(sv8in_04, axis=0)
std_sv8in_04 = np.std(sv8in_04, axis=0)
```

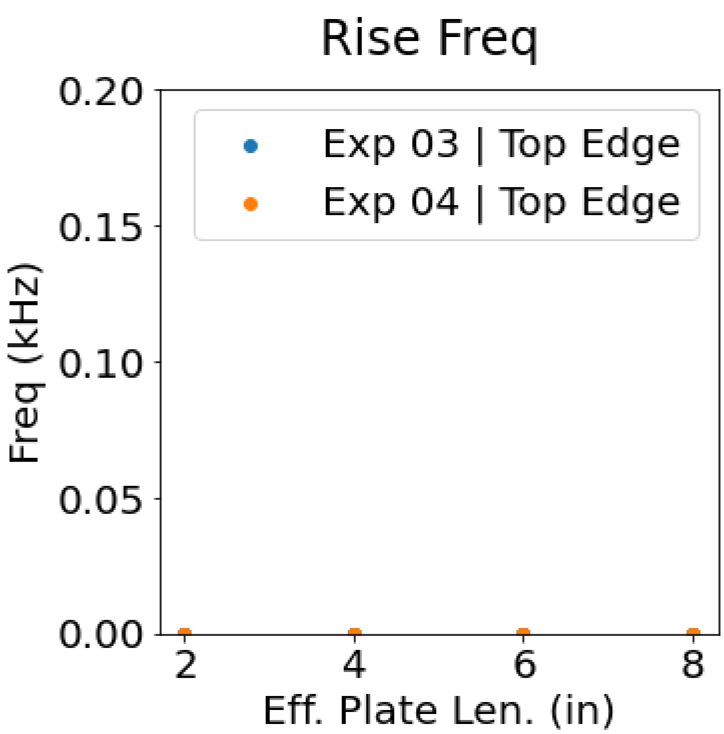
In [35]:

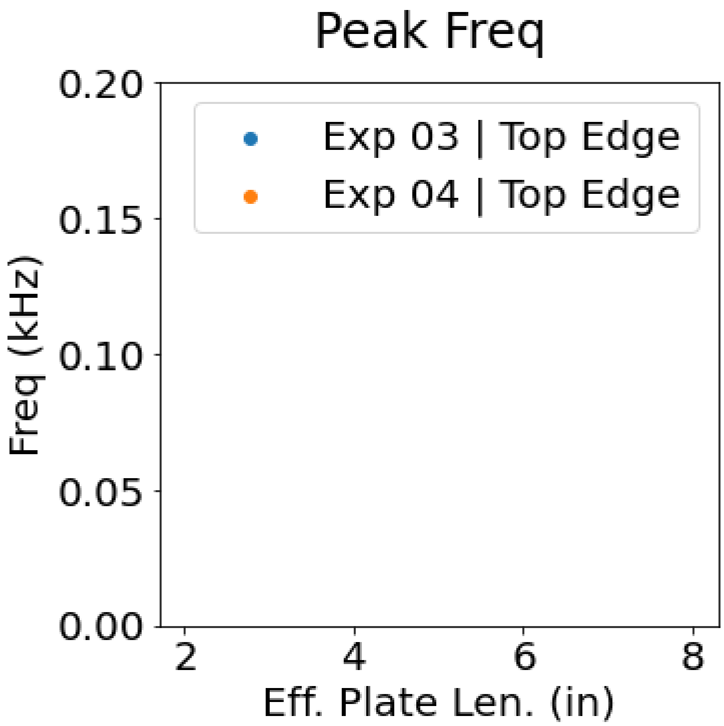
```
def plot_vs_length(title,b3,c3,b4,c4,a = ['2','4','6','8'],freq=False,pp=False):
    fig,spec2 = create_figure(title,columns=1,rows=1,width=5,height=5,default_font_size
                         ,tick_font_size=20,legend_font_size=20,axes_font_size=20,
                         title_font_size=22)

    ax = fig.add_subplot(spec2[0,0])

    plt.scatter(a, b3,label='Exp 03 | Top Edge')
    plt.scatter(a, b4,label='Exp 04 | Top Edge')

    plt.errorbar(a, b3, yerr=c3, fmt="o")
    plt.errorbar(a, b4, yerr=c4, fmt="o")

    if freq==True:
        plt.ylabel('Freq (kHz)')
    if pp == True:
        plt.ylabel('Norm. Area')

    plt.ylim([0.0,0.2])
    plt.legend()
    plt.xlabel('Eff. Plate Len. (in)')
    plt.show()
```

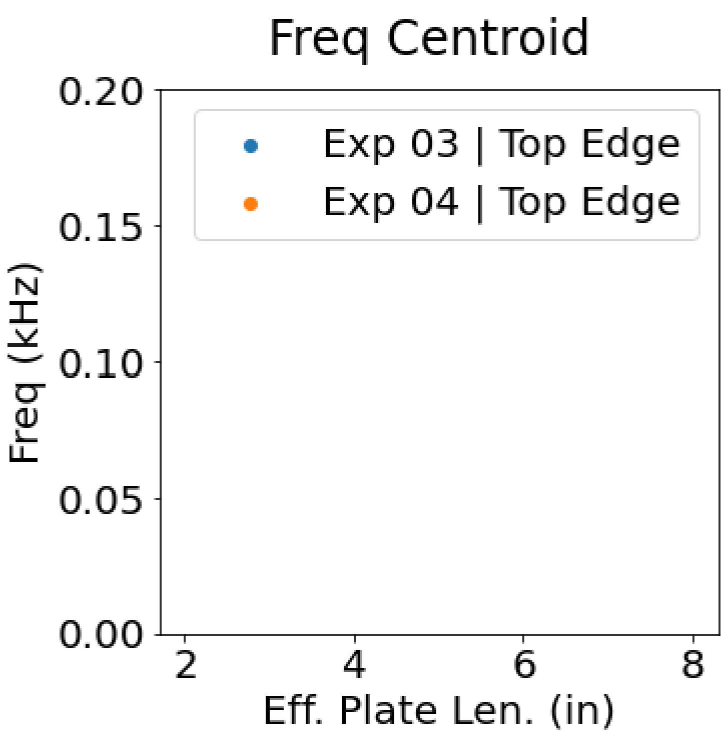
```
    return

features = ['Avg Freq','Reverb Freq','Rise Freq','Peak Freq','Freq Centroid','WPF',
           'PP1 (0-150kHz)','PP2 (150-300kHz)','PP3 (300-450kHz)','PP4 (450-600kHz)',
            'PP5 (600-900kHz)','PP6 (900-1200kHz)']

for idx,feature in enumerate(features):

    if idx < 6:

        b3 = [mean_sv2in_03[idx]/1000,mean_sv4in_03[idx]/1000,mean_sv6in_03[idx]/1000,m
        c3 = [std_sv2in_03[idx]/1000,std_sv4in_03[idx]/1000,std_sv6in_03[idx]/1000,std_

        b4 = [mean_sv2in_04[idx]/1000,mean_sv4in_04[idx]/1000,mean_sv6in_04[idx]/1000,m
        c4 = [std_sv2in_04[idx]/1000,std_sv4in_04[idx]/1000,std_sv6in_04[idx]/1000,std_

        plot_vs_length(feature,b3,c3,b4,c4,freq=True)

    else:

        b3 = [mean_sv2in_03[idx] ,mean_sv4in_03[idx] ,mean_sv6in_03[idx] ,mean_sv8in_03
        c3 = [std_sv2in_03[idx] ,std_sv4in_03[idx] ,std_sv6in_03[idx] ,std_sv8in_03[idx

        b4 = [mean_sv2in_04[idx] ,mean_sv4in_04[idx] ,mean_sv6in_04[idx] ,mean_sv8in_04
        c4 = [std_sv2in_04[idx] ,std_sv4in_04[idx] ,std_sv6in_04[idx] ,std_sv8in_04[idx

        plot_vs_length(feature,b3,c3,b4,c4,pp=True)
```

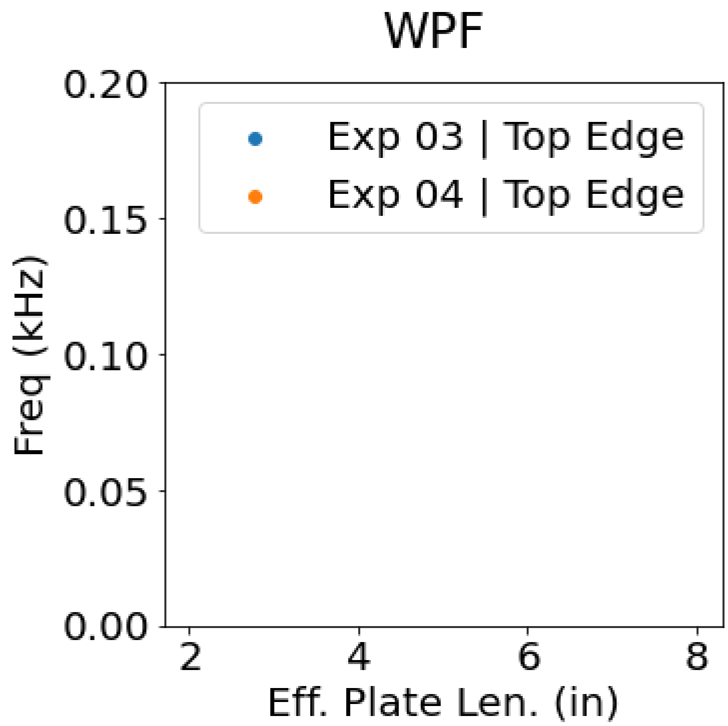


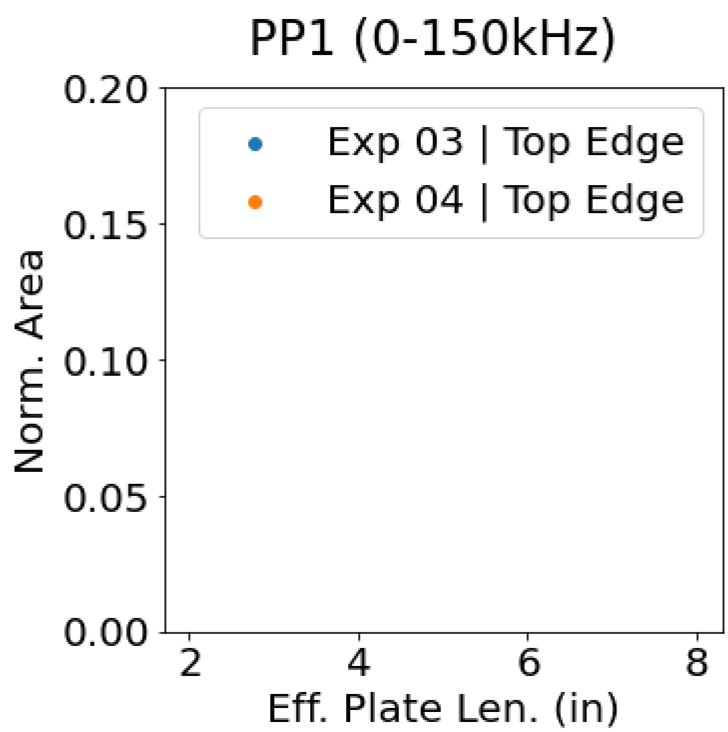

Reverb Freq
0.20
0.15
0.10
0.05
0.00
Freq (kHz)
Exp 03 | Top Edge
Exp 04 | Top Edge
2
4
6
8
Eff. Plate Len. (in)

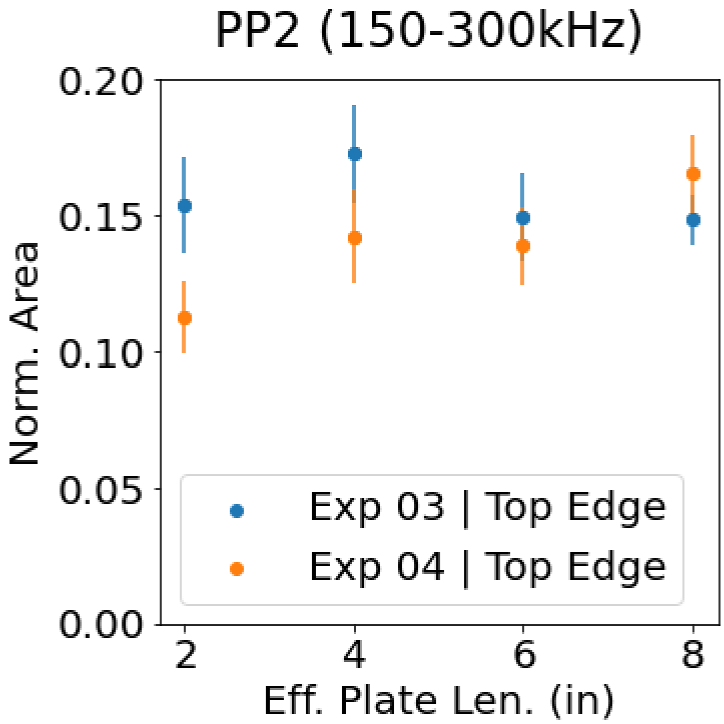

Rise Freq
0.20
0.15
0.10
0.05
0.00
Freq (kHz)
Exp 03 | Top Edge
Exp 04 | Top Edge
2
4
6
8
Eff. Plate Len. (in)

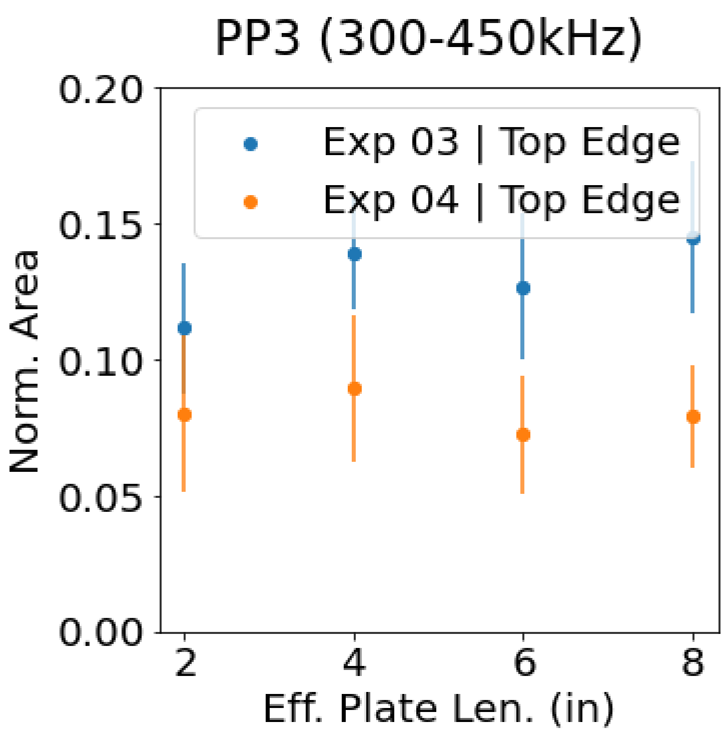

Peak Freq
Exp 03 | Top Edge
Exp 04 | Top Edge
Freq (kHz)
0.00
0.05
0.10
0.15
0.20
2
4
6
8
Eff. Plate Len. (in)

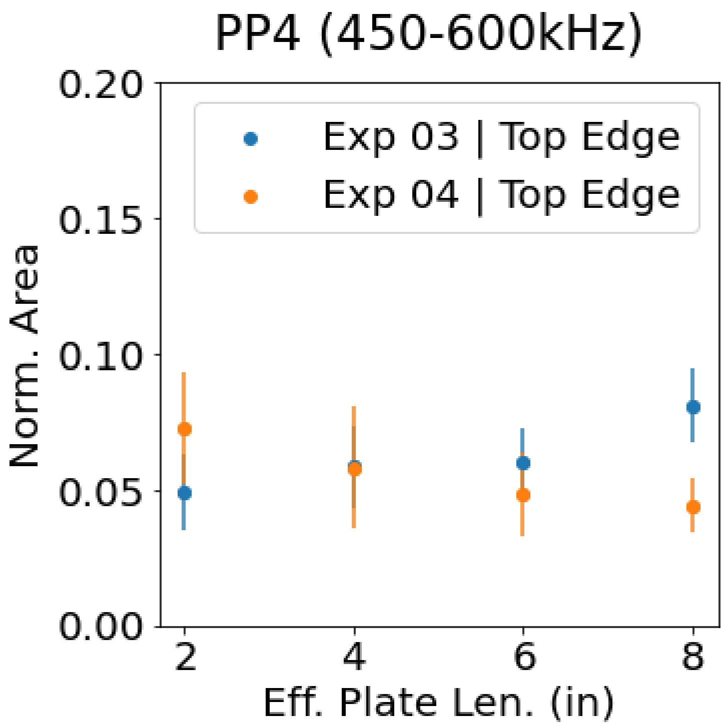

Freq Centroid
Exp 03 | Top Edge
Exp 04 | Top Edge
Freq (kHz)
0.00
0.05
0.10
0.15
0.20
2
4
6
8
Eff. Plate Len. (in)

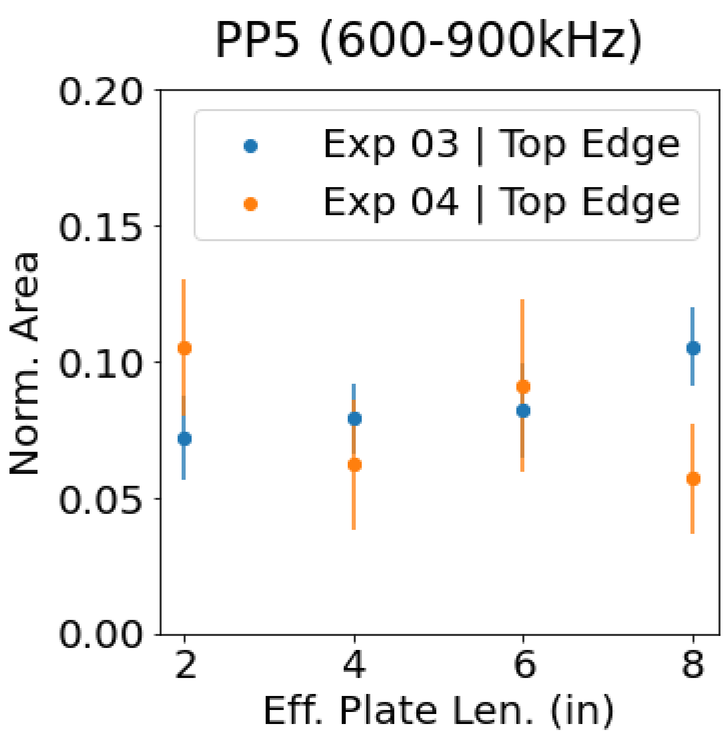

WPF
Exp 03 | Top Edge
Exp 04 | Top Edge
Freq (kHz)
0.20
0.15
0.10
0.05
0.00
2
4
6
8
Eff. Plate Len. (in)

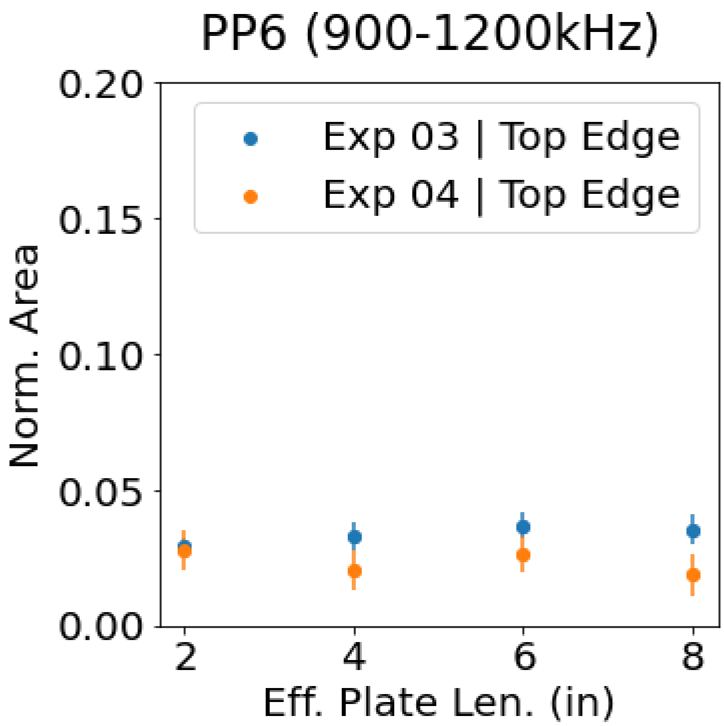

PP1 (0-150kHz)
Exp 03 | Top Edge
Exp 04 | Top Edge
Norm. Area
0.20
0.15
0.10
0.05
0.00
2
4
6
8
Eff. Plate Len. (in)


PP2 (150-300kHz)
Norm. Area
0.20
0.15
0.10
0.05
0.00
2
4
6
8
Eff. Plate Len. (in)
Exp 03 | Top Edge
Exp 04 | Top Edge


PP3 (300-450kHz)
Norm. Area
0.20
0.15
0.10
0.05
0.00
2
4
6
8
Eff. Plate Len. (in)
Exp 03 | Top Edge
Exp 04 | Top Edge


PP4 (450-600kHz)
Exp 03 | Top Edge
Exp 04 | Top Edge
Norm. Area
0.20
0.15
0.10
0.05
0.00
2
4
6
8
Eff. Plate Len. (in)


PP5 (600-900kHz)
Exp 03 | Top Edge
Exp 04 | Top Edge
Norm. Area
0.20
0.15
0.10
0.05
0.00
2
4
6
8
Eff. Plate Len. (in)


PP6 (900-1200kHz)
Exp 03 | Top Edge
Exp 04 | Top Edge
Norm. Area
0.20
0.15
0.10
0.05
0.00
2
4
6
8
Eff. Plate Len. (in)

```
In [ ]:
```