Load Data Tabel

In [1]:
```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt

# Load data from .csv table
ori_data = pd.read_csv('Original_Data.csv', encoding='gb18030')
fil_data = pd.read_csv('Filtered_Data.csv', encoding='gb18030')
```

Preview the Data

Original Data

In [2]:
```
ori_data.head()
```

Out[2]:

	X-Accelerate(m^2/s)	Y-Accelerate(m^2/s)	Z-Accelerate(m^2/s)
0	0.230000	-0.47	1.266136
1	0.040000	0.10	2.587760
2	0.080000	-0.43	2.856711
3	-0.490000	0.32	0.895768
4	-0.250000	0.30	3.588816

Filtered Data

In [3]:
```
fil_data.head()
```

Out[3]:

	X-Accelerate(m^2/s)	Y-Accelerate(m^2/s)	Z-Accelerate(m^2/s)
0	-0.078000	-0.036	2.239038
1	-0.078000	-0.036	2.239038
2	-0.078000	-0.036	2.239038
3	-0.078000	-0.036	2.239038
4	-0.078000	-0.036	2.239038

Extract Data from Tabel

In [4]:
```
# Original Data
ori_x = np.array(ori_data['X-Accelerate(m^2/s)'][:15000])
ori_y = np.array(ori_data['Y-Accelerate(m^2/s)'][:15000])
ori_z = np.array(ori_data['Z-Accelerate(m^2/s)'][:15000])
# Filtered Data
fil_x = np.array(fil_data['X-Accelerate(m^2/s)'][:15000])
fil_y = np.array(fil_data['Y-Accelerate(m^2/s)'][:15000])
fil_z = np.array(fil_data['Z-Accelerate(m^2/s)'][:15000])
# Steps
steps = np.array(ori_data['Y-Accelerate(m^2/s)'][15000:])
```

Transfer X-Data to Double Type

In [5]:
```
for i in range(15000):
    ori_x[i] = eval(ori_x[i])
    fil_x[i] = eval(fil_x[i])
```

Create the Cavanas

In [6]:
```
# Font Family Set
plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus']=False
f = plt.figure(dpi=300,
               figsize=(3.5, 2.5))
# Number of Samples We Selected to Display
display_n = 150
# Public Horizontal Data
X = np.array([a for a in range(1, display_n+1)])
```

```
<Figure size 1050x750 with 0 Axes>
```

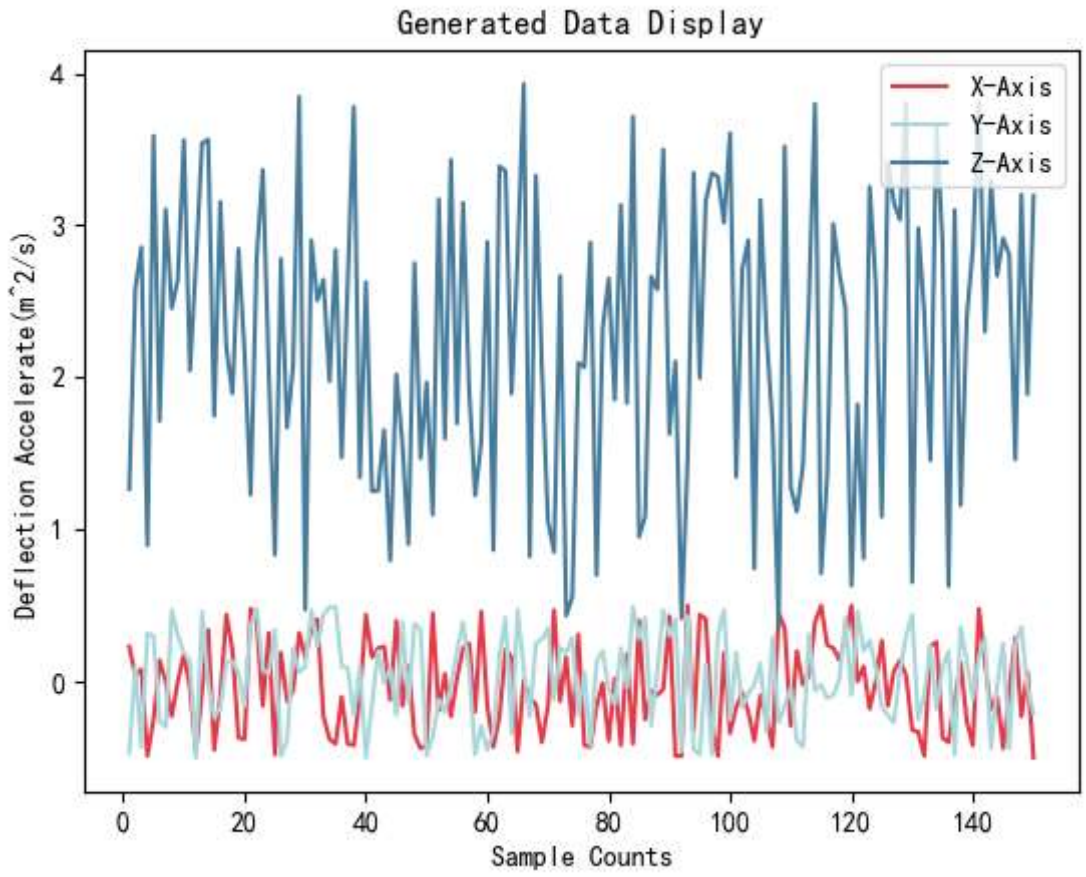
Check the Output Path

In [15]:
```
import pathlib

# Make sure the output exists
output = pathlib.Path('output')
if not output.exists():
    output.mkdir()
```

Display Original Data

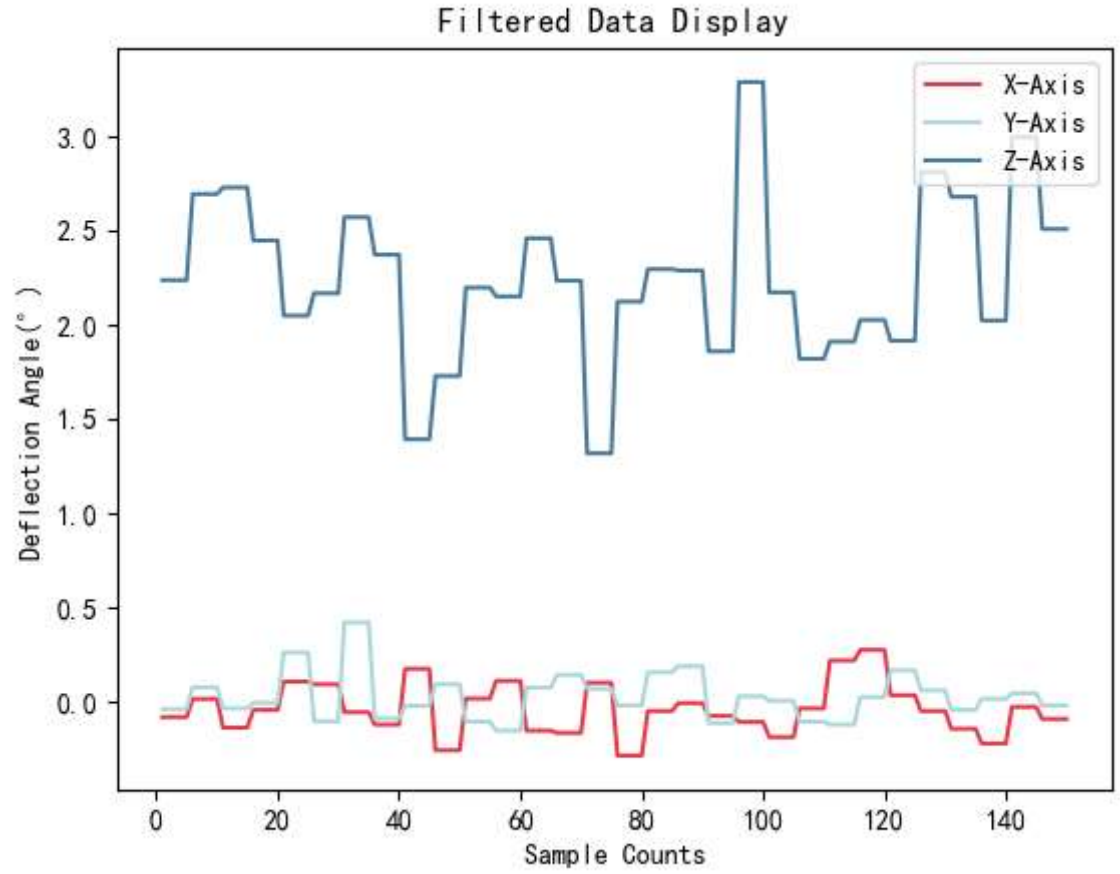
In [7]:
```
# x plot
plt.plot(X, ori_x[:display_n], color='#e73847', label='X-Axis')
# y plot
plt.plot(X, ori_y[:display_n], color='#a8d8db', label='Y-Axis')
# z plot
plt.plot(X, ori_z[:display_n], color='#457b9d', label='Z-Axis')
# Loc Show
plt.legend(loc="upper right")
# Title Set
plt.title("Generated Data Display")
# Axis Title
plt.xlabel("Sample Counts")
plt.ylabel("Deflection Accelerate(m^2/s)")
# Display the Figure
plt.show()
# Save the figure to local
plt.savefig('Original_Data.png')
```




```
<Figure size 640x480 with 0 Axes>
```

Display Filtered Data

In [8]:
```
# x plot
plt.plot(X, fil_x[:display_n], color='#e73847', label='X-Axis')
# y plot
plt.plot(X, fil_y[:display_n], color='#a8d8db', label='Y-Axis')
# z plot
plt.plot(X, fil_z[:display_n], color='#457b9d', label='Z-Axis')
# Loc Show
plt.legend(loc="upper right")
# Title Set
plt.title("Filtered Data Display")
# Axis Title
plt.xlabel("Sample Counts")
plt.ylabel("Deflection Angle(° )")
# Display the Figure
plt.show()
# Save the figure to local
plt.savefig('Filtered_Data.png')
```



<Figure size 640x480 with 0 Axes>

Get Dynamic Threshold

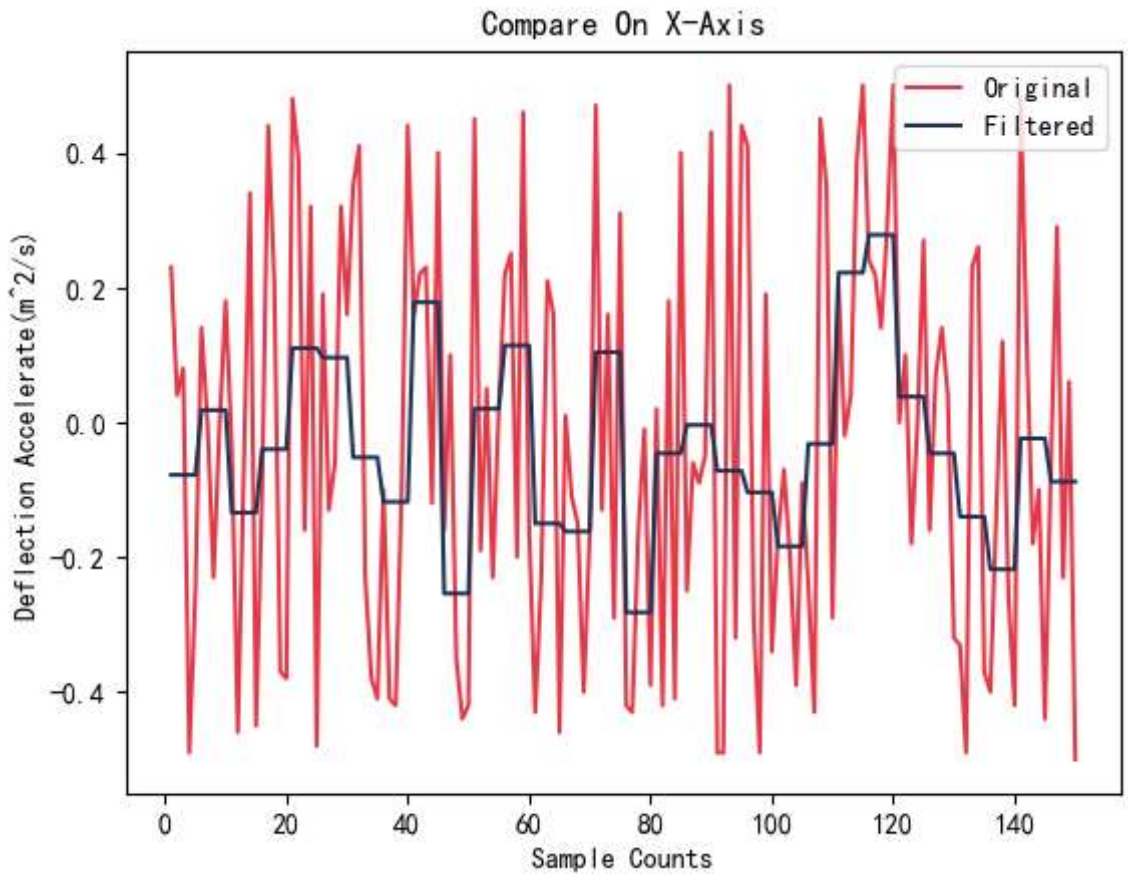
Load from .csv

In [9]:
```
# Load Tabel
thresholds_table = pd.read_csv('Thresholds.csv', encoding='gb18030')['Thresholds']
# Extent it
thresholds = []
for x in thresholds_table:
    for i in range(50):
        thresholds.append(x)
thresholds = np.array(thresholds)
# Primary Axis
primary_axis = pd.read_csv('Thresholds.csv', encoding='gb18030')['Primary Axis'][0]
```

Compare Filtered with Original

X-Axis

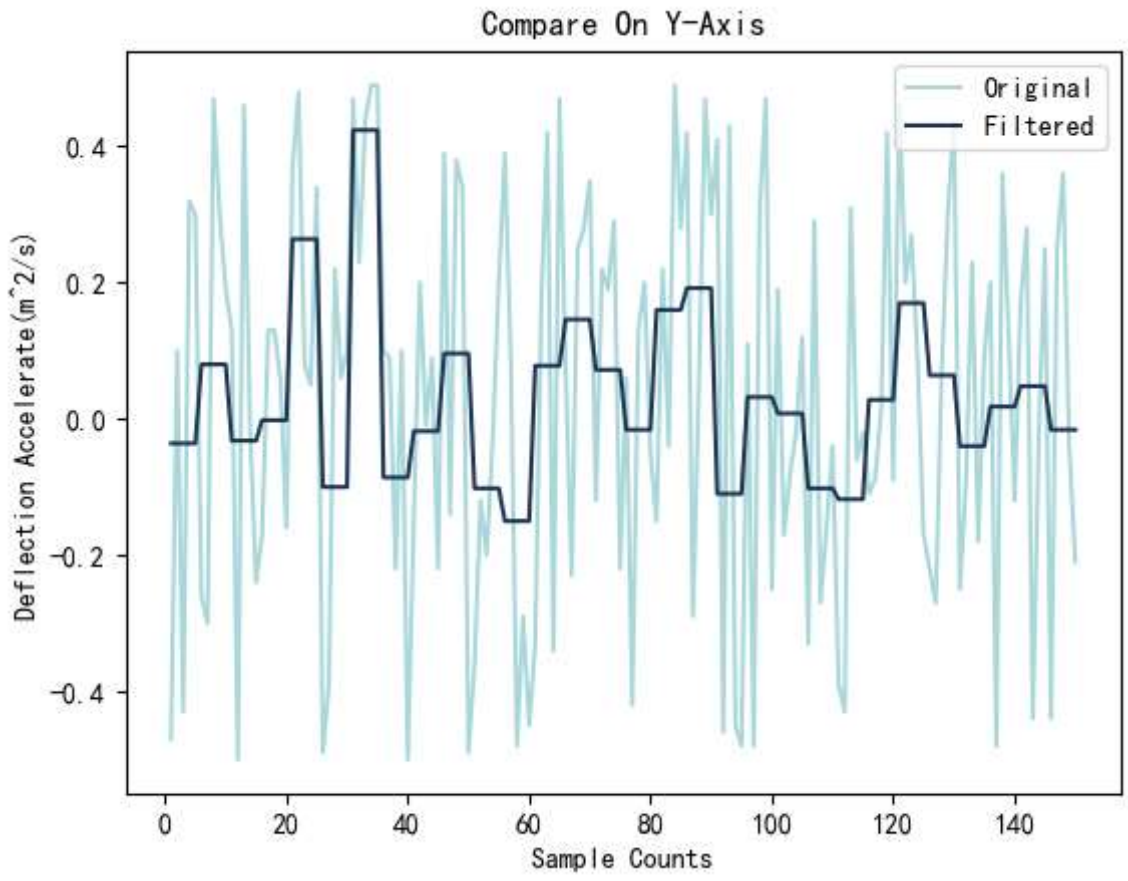
In [10]:
```
# Original
plt.plot(X, ori_x[:display_n], color='#e73847', label='Original')
# Filtered
plt.plot(X, fil_x[:display_n], color='#1d3557', label='Filtered')
# Threshold
if primary_axis == 0:
    plt.plot(X, thresholds[:display_n], color='#FEFE00', label='Thresholds')
# Loc Show
plt.legend(loc="upper right")
# Title Set
plt.title("Compare On X-Axis")
# Axis Title
plt.xlabel("Sample Counts")
plt.ylabel("Deflection Accelerate(m^2/s)")
# Display the Figure
plt.show()
# Save the figure to local
plt.savefig('Filtered_Data_X.png')
```



<Figure size 640x480 with 0 Axes>

Y-Axis

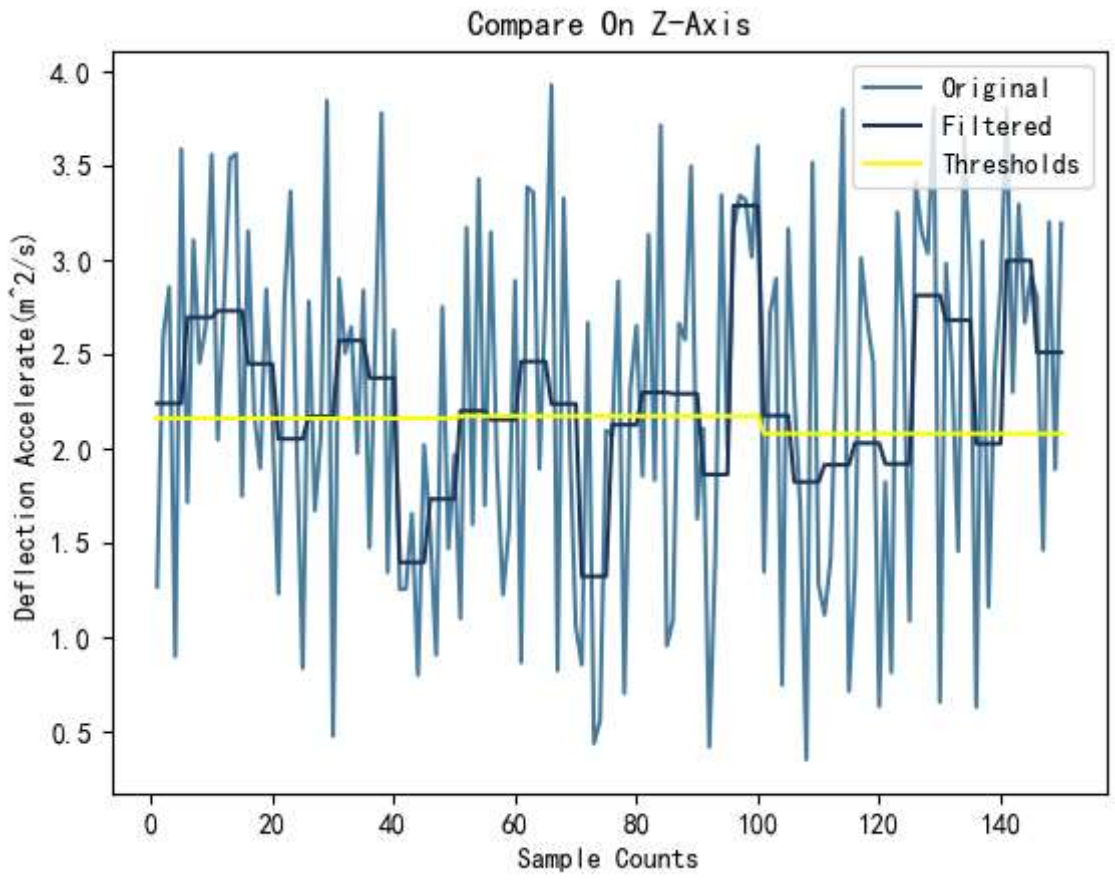
In [11]:
```
# Original
plt.plot(X, ori_y[:display_n], color='#a8d8db', label='Original')
# Filtered
plt.plot(X, fil_y[:display_n], color='#1d3557', label='Filtered')
# Threshold
if primary_axis == 1:
    plt.plot(X, thresholds[:display_n], color='#FEFE00', label='Thresholds')
# Loc Show
plt.legend(loc="upper right")
# Title Set
plt.title("Compare On Y-Axis")
# Axis Title
plt.xlabel("Sample Counts")
plt.ylabel("Deflection Accelerate(m^2/s)")
# Display the Figure
plt.show()
# Save the figure to local
plt.savefig('Filtered_Data_Y.png')
```



<Figure size 640x480 with 0 Axes>

Z-Axis

In [12]:
```
# Original
plt.plot(X, ori_z[:display_n], color='#457b9d', label='Original')
# Filtered
plt.plot(X, fil_z[:display_n], color='#1d3557', label='Filtered')
# Threshold
if primary_axis == 2:
    plt.plot(X, thresholds[:display_n], color='#FEFE00', label='Thresholds')
# Loc Show
plt.legend(loc="upper right")
# Title Set
plt.title("Compare On Z-Axis")
# Axis Title
plt.xlabel("Sample Counts")
plt.ylabel("Deflection Accelerate(m^2/s)")
# Display the Figure
plt.show()
# Save the figure to local
plt.savefig('Filtered_Data_Z.png')
```



<Figure size 640x480 with 0 Axes>

In [ ]:
```
1
```