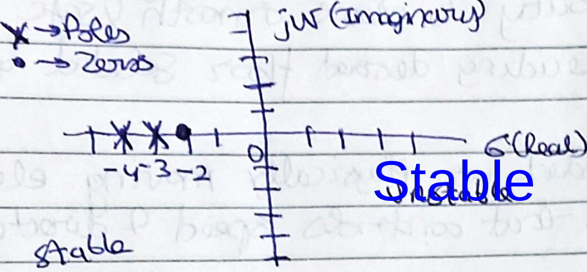
Assignment 2:- Poles, Zeroes, Stability

→ **Algorithm:-**
1) Start
2) Input Transfer Function
3) Find poles and zeros
4) Determine Stability
5) Plot poles & zeros on S-plane
6) End

Example:- $G(s) = \dfrac{s+2}{s^2+7s+12} = \dfrac{s+2}{(s+4)(s+3)}$

Zeros:- $s+2=0$; $s=-2$

Poles:- $(s+4)(s+3)=0$; $s=-4$, $s=-3$

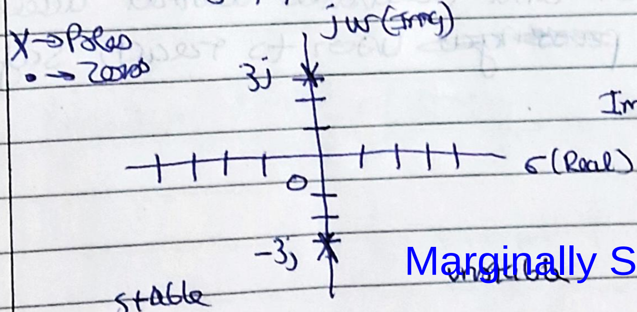


Stable

S-plane

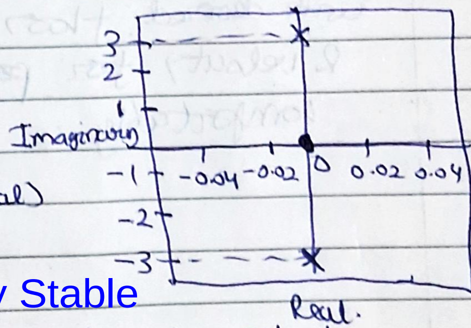
Output confirmed as per Python-codes.

$G(s) = \dfrac{s}{s^2+9}$ → Poles $(s^2+9)=0$, $s = \pm 3j$



Marginally Stable

S-plane



in python output

→ Example $G(s) = \dfrac{1}{s-4}$

Poles:- $s-4=0 \rightarrow s=4$

$\boxed{\text{Unstable}}$

Zeroes, Poles and Stability

Python

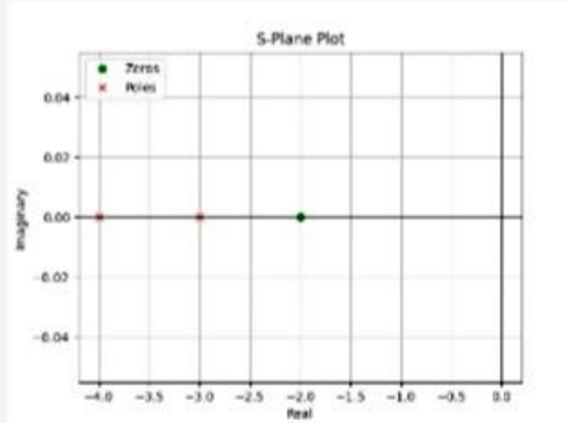
Run Save

```python
import numpy as np
import matplotlib.pyplot as plt

# Define the transfer function representing the Mechatronics System
numerator = [1, 2]  # Coefficients of numerator
denominator = [1, 7, 12]  # Coefficients of denominator

# Find poles and zeros
zeros = np.roots(numerator)
poles = np.roots(denominator)

# Determine stability
all_real_negative = all(np.real(p) < 0 for p in poles)
all_real_nonpositive = all(np.real(p) <= 0 for p in poles)
any_imaginary = any(np.imag(p) != 0 for p in poles)

if all_real_negative:
    stability = "Stable"
elif all_real_nonpositive and any_imaginary:
    stability = "Marginally Stable"
else:
    stability = "Unstable"

# Plot poles and zeros on the s-plane
plt.plot(np.real(zeros), np.imag(zeros), 'go', label='Zeros')
plt.plot(np.real(poles), np.imag(poles), 'rx', label='Poles')
plt.axhline(0, color='black',linewidth=0.5)
plt.axvline(0, color='black',linewidth=0.5)
plt.xlabel('Real')
plt.ylabel('Imaginary')
plt.title('S-Plane Plot')
plt.grid()
plt.legend()
plt.show()

# Output stability
print("System is", stability)

```

Program input

Output

```
System is Stable

[Execution complete with exit code
```



Zeroes, Poles and Stability

Python

Run

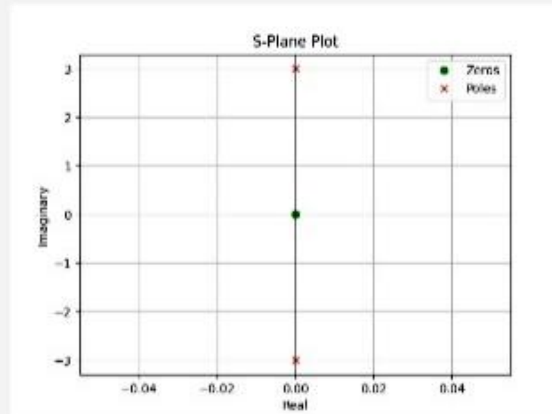
Save

```python
import numpy as np
import matplotlib.pyplot as plt

# Define the transfer function representing the Mechatronics System
numerator = [1, 0]  # Coefficients of numerator
denominator = [1, 0, 9]  # Coefficients of denominator

# Find poles and zeros
zeros = np.roots(numerator)
poles = np.roots(denominator)

# Determine stability
all_real_negative = all(np.real(p) < 0 for p in poles)
all_real_nonpositive = all(np.real(p) <= 0 for p in poles)
any_imaginary = any(np.imag(p) != 0 for p in poles)

if all_real_negative:
    stability = "Stable"
elif all_real_nonpositive and any_imaginary:
    stability = "Marginally Stable"
else:
    stability = "Unstable"

# Plot poles and zeros on the s-plane
plt.plot(np.real(zeros), np.imag(zeros), 'go', label='Zeros')
plt.plot(np.real(poles), np.imag(poles), 'rx', label='Poles')
plt.axhline(0, color='black',linewidth=0.5)
plt.axvline(0, color='black',linewidth=0.5)
plt.xlabel('Real')
plt.ylabel('Imaginary')
plt.title('S-Plane Plot')
plt.grid()
plt.legend()
plt.show()

# Output stability
print("System is", stability)

```

Program input

Output

```
System is Marginally Stable

[Execution complete with exit code (
```



Zeroes, Poles and Stability

Python

Run

Save

```python
import numpy as np
import matplotlib.pyplot as plt

# Define the transfer function representing the Mechatronics System
numerator = [1]  # Coefficients of numerator
denominator = [1, -4]  # Coefficients of denominator

# Find poles and zeros
zeros = np.roots(numerator)
poles = np.roots(denominator)

# Determine stability
all_real_negative = all(np.real(p) < 0 for p in poles)
all_real_nonpositive = all(np.real(p) <= 0 for p in poles)
any_imaginary = any(np.imag(p) != 0 for p in poles)

if all_real_negative:
    stability = "Stable"
elif all_real_nonpositive and any_imaginary:
    stability = "Marginally Stable"
else:
    stability = "Unstable"

# Plot poles and zeros on the s-plane
plt.plot(np.real(zeros), np.imag(zeros), 'go', label='Zeros')
plt.plot(np.real(poles), np.imag(poles), 'rx', label='Poles')
plt.axhline(0, color='black',linewidth=0.5)
plt.axvline(0, color='black',linewidth=0.5)
plt.xlabel('Real')
plt.ylabel('Imaginary')
plt.title('S-Plane Plot')
plt.grid()
plt.legend()
plt.show()

# Output stability
print("System is", stability)
```

Program input

Output

```
System is Unstable

[Execution complete with exit code (
```