2.1\2.1.py

```python
#%% Import libraries
import numpy as np
import matplotlib.pyplot as plt

def derivative(x, y, A_sigma):
    return A_sigma @ np.array([x, y])

def classify_stability(eigenvalues):
    real_parts = np.real(eigenvalues)
    imag_parts = np.imag(eigenvalues)

    if all(real_parts < 0):
        return "Stable Node"
    elif all(real_parts > 0):
        return "Unstable Node"
    elif real_parts[0] * real_parts[1] < 0:
        return "Saddle Point"
    elif imag_parts[0] != 0:
        if real_parts[0] == 0:
            return "Center"
        elif real_parts[0] < 0:
            return "Stable Focus"
        else:
            return "Unstable Focus"
    else:
        return "Other"

#%% Run and plot
sigmaList = [-1, 0, 1]
x = np.linspace(-2, 2, 20)
y = np.linspace(-2, 2, 20)
X, Y = np.meshgrid(x, y)

# Create subplots: 1 row, 3 columns
fig, axes = plt.subplots(1, 3, figsize=(18, 6))
fig.suptitle("Trajectories for Different Sigma Values")

for idx, sigma in enumerate(sigmaList):
    A_sigma = np.array([[sigma + 3, 4], [-9/4, sigma - 3]])
    dX, dY = np.zeros(X.shape), np.zeros(Y.shape)

    # Calculate derivatives
    for i in range(X.shape[0]):
        for j in range(X.shape[1]):
            dx, dy = derivative(X[i, j], Y[i, j], A_sigma)
            magnitude = np.sqrt(dx**2 + dy**2)
            if magnitude > 0:
                dX[i, j], dY[i, j] = dx / magnitude, dy / magnitude

    # Plot vector field in the corresponding subplot
    ax = axes[idx]
```

```
    ax.streamplot(X, Y, dX, dY)
    ax.set_title(f"Sigma = {sigma}")
    ax.set_xlabel("x")
    ax.set_ylabel("y")
    ax.grid()

    # Stability analysis
    eigenvalues, _ = np.linalg.eig(A_sigma)
    stability = classify_stability(eigenvalues)
    print(f"Sigma = {sigma}: {stability}")

# Adjust layout
plt.tight_layout()
plt.show()
```