```python
import numpy as np


# Given channel matrix H
m=3;
H = np.array([[3, 0, 8],
              [0, 1, 0],
              [4, 0, 6]])

# Calculate eigenvalues of H*H^H
singularvalues = np.linalg.eigvals(H @ H.T)

# Print the singular values
print("singular values of H*H^H:")
print(singularvalues)




def svd_channel_matrix(H):
    U, D, Vh = np.linalg.svd(H)
    return U, D, Vh.T  # Transpose Vh to match MATLAB's convention

# Given channel matrix H
H = np.array([[3, 0, 8],
              [0, 1, 0],
              [4, 0, 6]])

# Call the function
U, D, V = svd_channel_matrix(H)

# Print the results
print("U matrix:")
print(U)
print("\nS matrix (singular values):")
print(np.diag(D))
print("\nV matrix:")
print(V)




def compute_capacity_multiplexing(H, sigma_n_squared, P_total, channel_knowledge=True):
    U, s, Vh = svd_channel_matrix(H)
    if channel_knowledge:
        capacity = (np.log2(1 + P_total*(max(singularvalues)) / sigma_n_squared))
    else:
        z=P_total*singularvalues
        o=m*sigma_n_squared
        capacity = np.sum(np.log2(1 + z/o))
    return capacity

def compute_capacity_diversity(H, sigma_n_squared, P_total, channel_knowledge=True):
    U, s, Vh = svd_channel_matrix(H)
    if channel_knowledge:

        capacity = (np.log2(1 + P_total*(max(singularvalues)) / sigma_n_squared))
    else:

        f=np.sum(singularvalues)
        x=P_total*f
        y=m*sigma_n_squared

        capacity = (np.log2(1 + x/y))
    return capacity

# Example usage
H = np.array([[3, 0, 8], [0, 1, 0], [4, 0, 6]])
sigma_n_squared = 10**(3/10)  # Convert dB to linear scale
P_total = 10**((-1.75)/10)    # Convert dB to linear scale

# Multiplexing with channel knowledge
capacity_multiplexing_with_ck = compute_capacity_multiplexing(H, sigma_n_squared, P_total)
print("Multiplexing capacity with channel knowledge:", capacity_multiplexing_with_ck)

# Multiplexing without channel knowledge
capacity_multiplexing_without_ck = compute_capacity_multiplexing(H, sigma_n_squared, P_total, channel_knowledge=False)
```

```
print("Multiplexing capacity without channel knowledge:", capacity_multiplexing_without_ck)

# Diversity with channel knowledge
capacity_diversity_with_ck = compute_capacity_diversity(H, sigma_n_squared, P_total)
print("Diversity capacity with channel knowledge:", capacity_diversity_with_ck)

# Diversity without channel knowledge
capacity_diversity_without_ck = compute_capacity_diversity(H, sigma_n_squared, P_total, channel_knowledge=False)
print("Diversity capacity without channel knowledge:", capacity_diversity_without_ck)
```