```
import numpy as np

# Function to convert text to numerical values (A=0, B=1, ..., Z=25)
def text_to_numbers(text):
    return [ord(char) - ord('A') for char in text.upper()]

# Function to convert numerical values back to text
def numbers_to_text(numbers):
    return ''.join(chr(num + ord('A')) for num in numbers)

# Function to encrypt plaintext using Hill Cipher
def hill_encrypt(plaintext, key_matrix):
    # Convert plaintext to numbers
    text_vector = text_to_numbers(plaintext)

    # Reshape text vector to fit matrix multiplication
    text_vector = np.array(text_vector).reshape(-1, len(key_matrix))

    # Perform matrix multiplication and mod 26
    encrypted_vector = np.dot(text_vector, key_matrix) % 26

    # Convert numbers back to text
    encrypted_text = numbers_to_text(encrypted_vector.flatten())
    return encrypted_text

# Function to decrypt ciphertext using Hill Cipher
def hill_decrypt(ciphertext, key_matrix):
```

```
    # Convert ciphertext to numbers
    text_vector = text_to_numbers(ciphertext)

    # Compute inverse of the key matrix modulo 26
    det = int(round(np.linalg.det(key_matrix)))
    det_inv = pow(det, -1, 26)  # Modular inverse of determinant

    # Adjugate matrix and mod 26
    key_inv = np.round(det_inv * np.linalg.inv(key_matrix) * det).astype(int) % 26

    # Reshape text vector to fit matrix multiplication
    text_vector = np.array(text_vector).reshape(-1, len(key_matrix))

    # Perform matrix multiplication and mod 26
    decrypted_vector = np.dot(text_vector, key_inv) % 26

    # Convert numbers back to text
    decrypted_text = numbers_to_text(decrypted_vector.flatten())
    return decrypted_text

# Example usage
key_matrix = np.array([[6, 24], [1, 13]])  # 2x2 key matrix
plaintext = "HI"

# Encrypting
ciphertext = hill_encrypt(plaintext, key_matrix)
print(f"Encrypted text: {ciphertext}")
```

```
# Decrypting
decrypted_text = hill_decrypt(ciphertext, key_matrix)
print(f"Decrypted text: {decrypted_text}")
```