Problema 1

Implementar el algoritmo de *Gram-Schmidt* modificado 8.1 del Trefet-hen (p. 58) para generar la descomposición *QR*

Algorithm 1: Modified Gram-Schmidt (Trefethen pag. 58)

```
for idx in range(0, n):
    # Normalize ortogonal vector
    R[idx, idx] = np.sqrt(Q[:, idx].dot(Q[:, idx]))
    Q[:, idx] /= R[idx, idx]

    # Substract projection to q_i ortonormal vector
    for jdx in range(idx + 1, n):
        R[idx, jdx] = Q[:, idx].dot(Q[:, jdx])
        Q[:, jdx] -= R[idx, jdx] * Q[:, idx]
```

Problema 2

Implementar el algoritmo que calcula el estimador de mínimos cuadra-dos en una regresión usando la descomposición *QR*.

Algorithm 2: Least Squares estimator.

```
def least_squares_estimator(X, Y):
    # Get QR decomposition of data matrix
    (Q, R) = qr_factorization(X)

    # Transform y vector
    Y_prime = Q.T @ Y.T

    # Solve system R * beta = y_prime
    beta = backward_substitution(R, Y_prime.T)

    return beta
```

Problema 3

Generar Y **compuesto de** $y_i = sen(x_i) + \epsilon_i$ **donde** ϵ_i $N(0, \sigma)$ **con** $sigma = 0{,}11$ **para** $x_i = \frac{4\pi i}{n}$ **para** $i = 1, ..., n$**.**

Hacer un ajuste de mínimos cuadrados a Y, con descomposición QR, ajustando un polinomio de grado $p-1$.

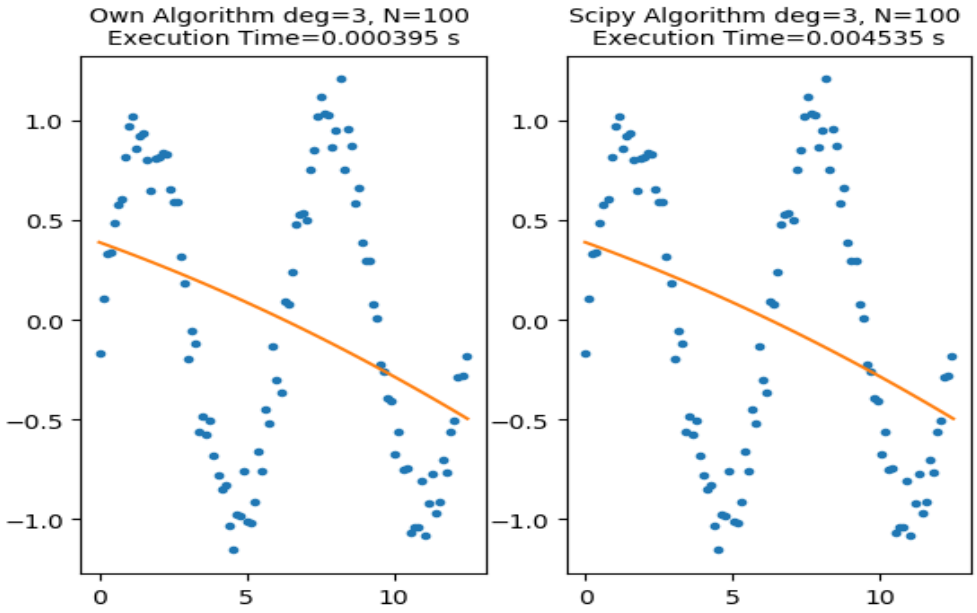

Figura 1: Comparison

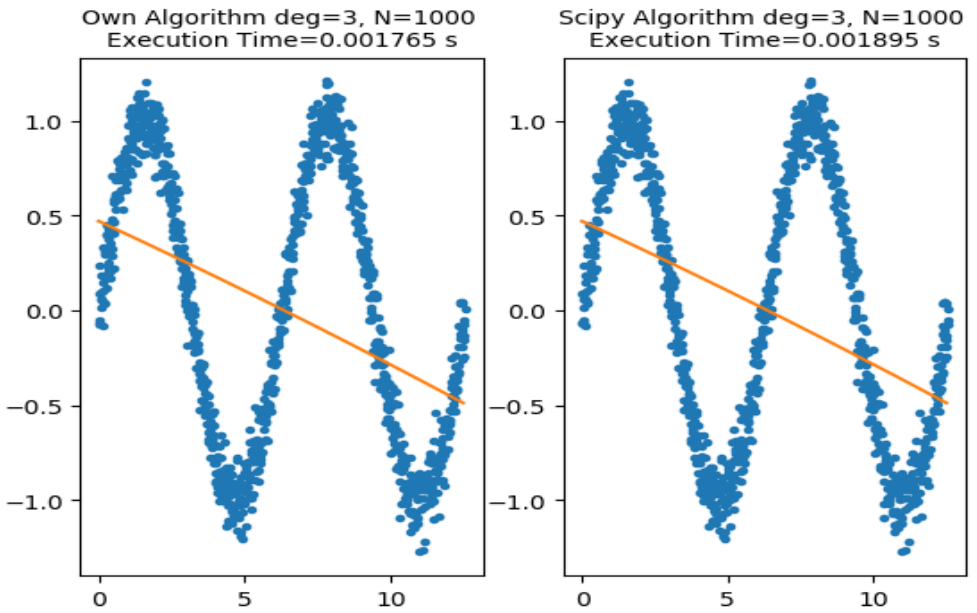

Figura 2: Comparison

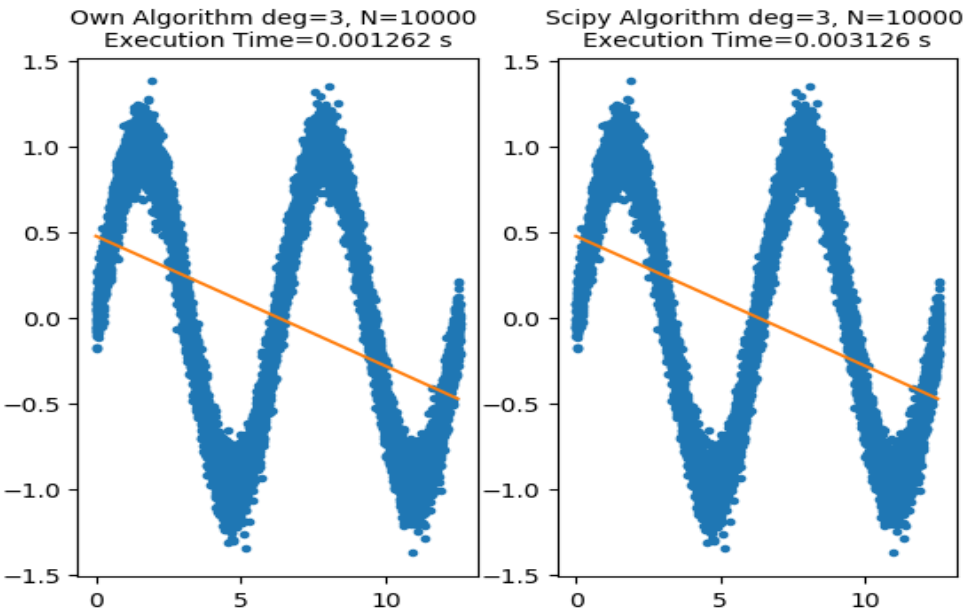

Figura 3: Comparison

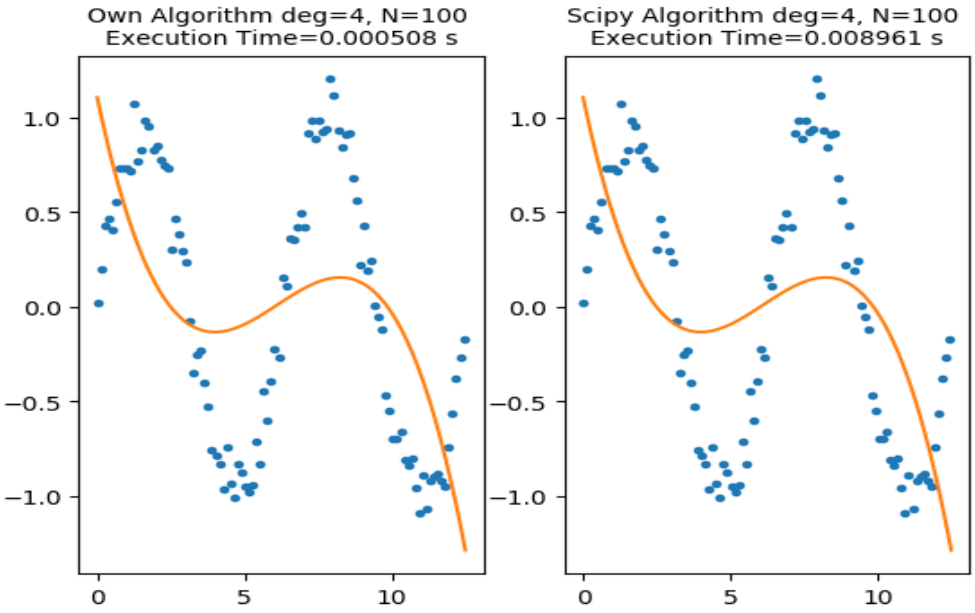

Figura 4: Comparison

Problema 4

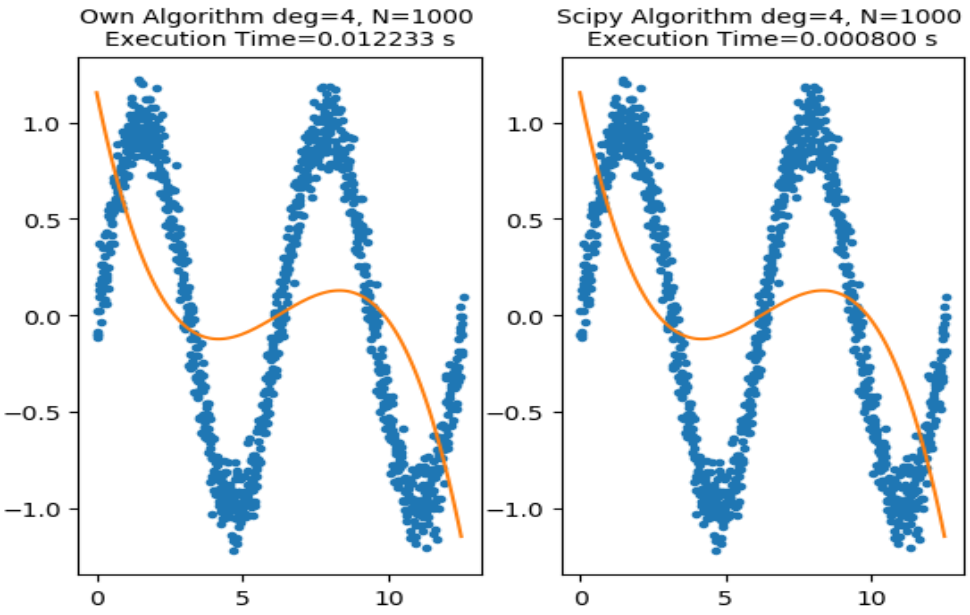

Figura 5: Comparison

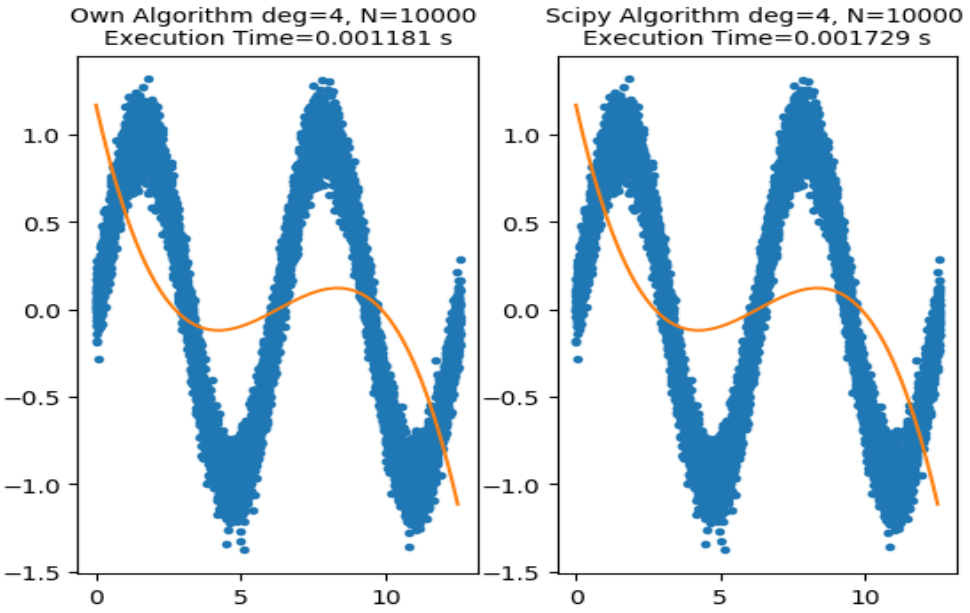

Figura 6: Comparison

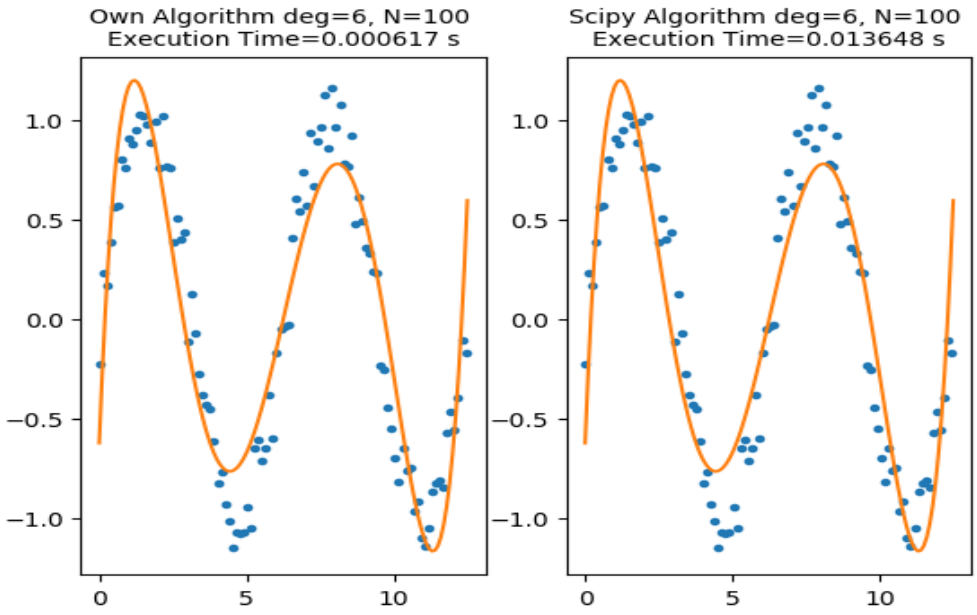

Figura 7: Comparison

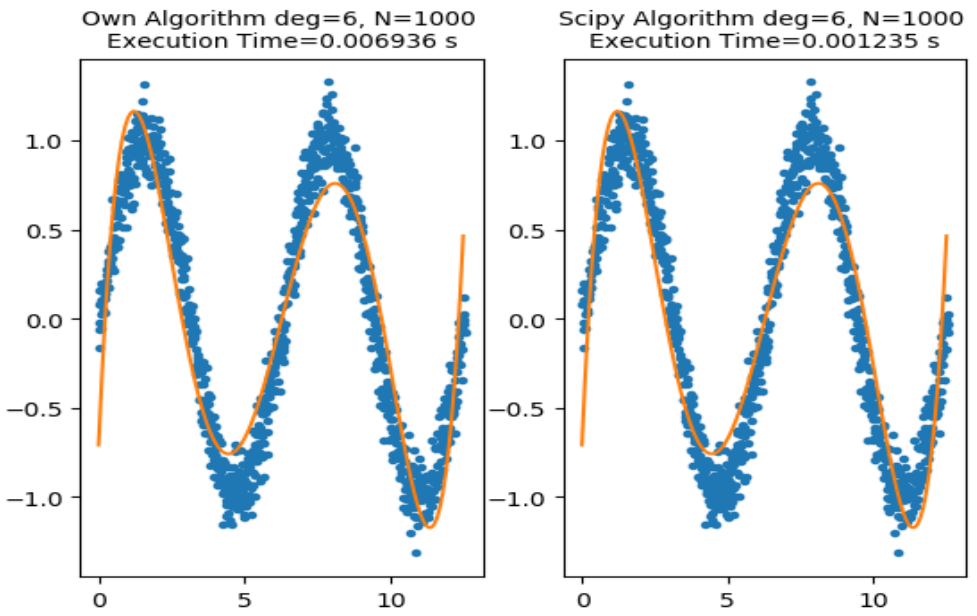

Figura 8: Comparison

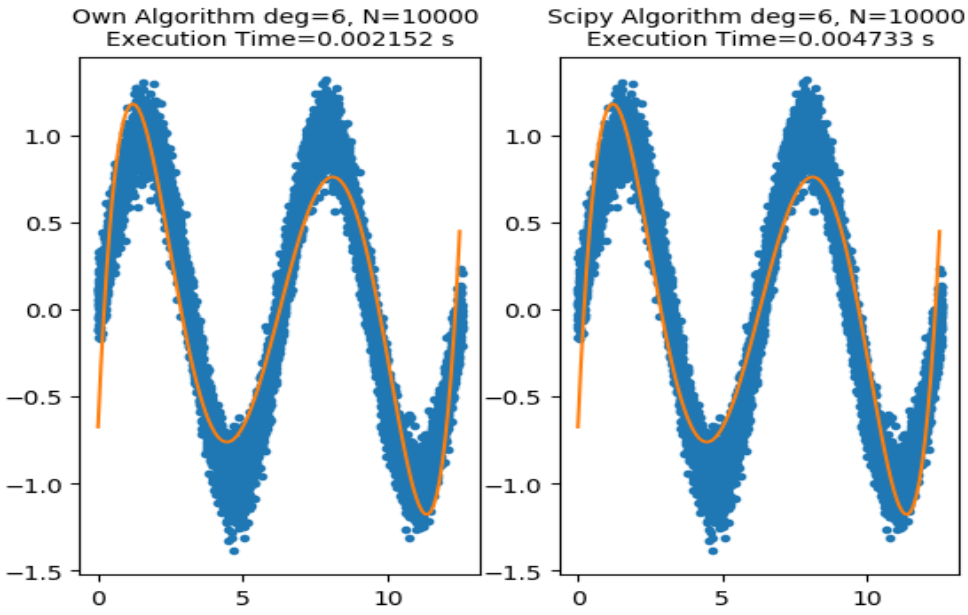

Figura 9: Comparison

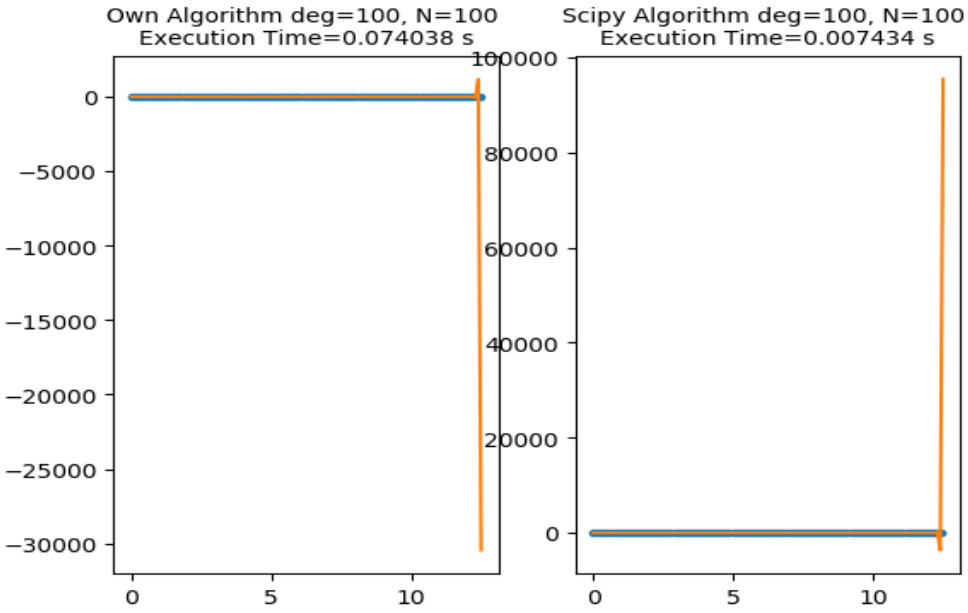

Figura 10: Comparison

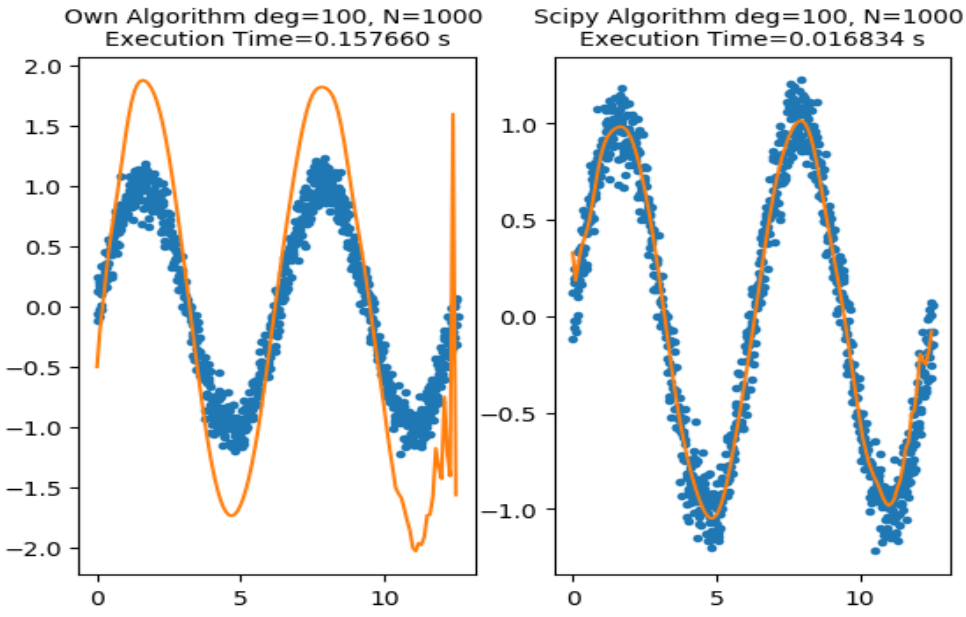

Figura 11: Comparison

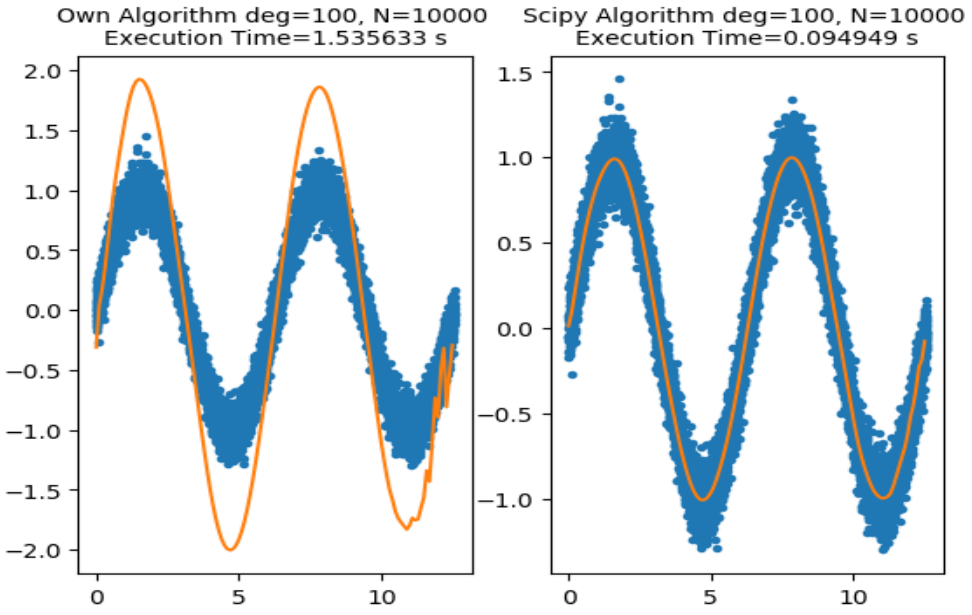

Figura 12: Comparison