1. **[20 MARKS]** Solve the problem of fitting a polynomial $p(x) = \sum_{i=0}^{d} c_i x^{i-1}$ of degree d to data points $(x_i, y_i)$, $i = 1, \ldots, m$, in the plane by the method of normal equations and QR decomposition. Choose the degree of the polynomial to be d=5 and then d=15, choose the interval $x \in [-1, 1]$, discretize it using $N = 10$ or $N = 20$ points.

Such polynomial fitting leads to the equation $Ac = y$, where $A$ is the Vandermonde matrix, $c$ is the vector of coefficients, and $y$ is the vector of data points. The normal equations are given by:

$$A^T Ac = A^T y$$

The solution to the normal equations is given by:

$$c = (A^T A)^{-1} A^T y$$

By using the QR decomposition, the matrix $A^T A$ is factorized as follows:

$$A^T A = QR$$

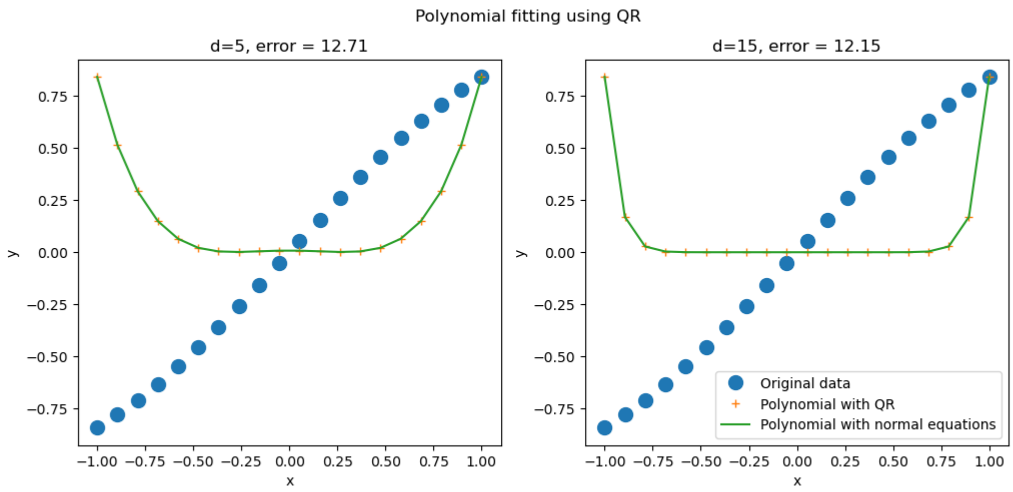
where $Q$ is an orthogonal matrix and $R$ is an upper triangular matrix. The solution to the normal equations is given by:

$$c = R^{-1} Q^T A^T y$$

The Python script to solve the problem of fitting a polynomial $p(x) = \sum_{i=0}^{d} c_i x^{i-1}$ of degree $d$ to data points $(x_i, y_i)$, $i = 1, \ldots, m$, in the plane by the method of normal equations and QR decomposition is given as follows:

In [7]:

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

fig, axs = plt.subplots(nrows=1, ncols=2, figsize=(12, 5))
# Define the degree of the polynomial
d = 5

# Define the interval
a = -1
b = 1

# Define the number of points
N = 20

# Define the data points
x = np.linspace(a, b, N)
y = np.sin(x)

# Define the Vandermonde matrix
A = np.vander(x, d + 1)

# Define the vector of data points
y = y.reshape(N, 1)

# QR decomposition of the matrix A^{T}A
Q, R = np.linalg.qr(A.T @ A)

# Solve the normal equations
c1 = np.linalg.solve(R, Q.T @ A.T @ y)

# Solve the normal equations
c2 = np.linalg.solve(A.T @ A, A.T @ y)

# plot both solutions

# plot original data
axs[0].plot(x, y, 'o', label='Original data', markersize=10)

# plot the polynomial with QR
axs[0].plot(x, np.polyval(c1[::-1], x), '+', label='Polynomial with QR')

# plot the polynomial with normal equations
axs[0].plot(x, np.polyval(c2[::-1], x), '-', label='Polynomial with normal equations')

axs[0].set_title(f"d=5, error = {round(np.linalg.norm(y-np.polyval(c1[::-1], x)), 2)}")

axs[0].set_xlabel("x")
axs[0].set_ylabel("y")


# Define the degree of the polynomial
d = 15


# Define the Vandermonde matrix
A = np.vander(x, d + 1)

# Define the vector of data points
y = y.reshape(N, 1)

# QR decomposition of the matrix A^{T}A
Q, R = np.linalg.qr(A.T @ A)

# Solve the normal equations
c1 = np.linalg.solve(R, Q.T @ A.T @ y)

# Solve the normal equations
c2 = np.linalg.solve(A.T @ A, A.T @ y)

# plot both solutions

# plot original data
axs[1].plot(x, y, 'o', label='Original data', markersize=10)

# plot the polynomial with QR
axs[1].plot(x, np.polyval(c1[::-1], x), '+', label='Polynomial with QR')

# plot the polynomial with normal equations
axs[1].plot(x, np.polyval(c2[::-1], x), '-', label='Polynomial with normal equations')

axs[1].set_title(f"d=15, error = {round(np.linalg.norm(y-np.polyval(c1[::-1], x)), 2)}")


axs[1].set_xlabel("x")
axs[1].set_ylabel("y")

plt.legend()

plt.suptitle("Polynomial fitting using QR")
plt.show()
# fig.savefig('poly.png', bbox_inches='tight')
```



Polynomial fitting using QR
d=5, error = 12.71
d=15, error = 12.15
Original data
Polynomial with QR
Polynomial with normal equations
x
y


In [ ]: