Problem 2

This past week we have discussed many forms of interpolation. Polynomial interpolation gives us a simple machinery to fit a mathematical model, a polynomial in this case, for a series of datapoints. Lagrange polynomials provide M polynomial basis functions to fit data of M + 1 points without having to solve a linear system of equations, as the polynomial fit is unique under this condition. Trigonometric polynomials can also be used for interpolation and the infinite degree trigonometric polynomial is known as the Fourier Series, which has seen use in many different fields for fitting data and approximating functions as sum of sinusoids. Splines are a useful tool that can provide a higher fidelity fit, as they fit inidividual polynomials on the data intervals. They also ensure the stability of the fit by considering the derivatives at the points where the individual polynomials intersect.

Interpolation is a very useful and practical tool. During my undergrad aerospace labs, we would use linear and quadratic polynomial interpolations to estimate system parameters. Given a physics derived model of lift/drag forces and velocity, these parameters would be determined based on a polynomial fitted on the data obtained from experiments. Another use of interpolation is in the domain of trajectory generation. Often times you have a set of high level waypoints, but to generate a smooth upsampled sequence of points so that a controller can track the trajectory requires the use of spline interpolation.

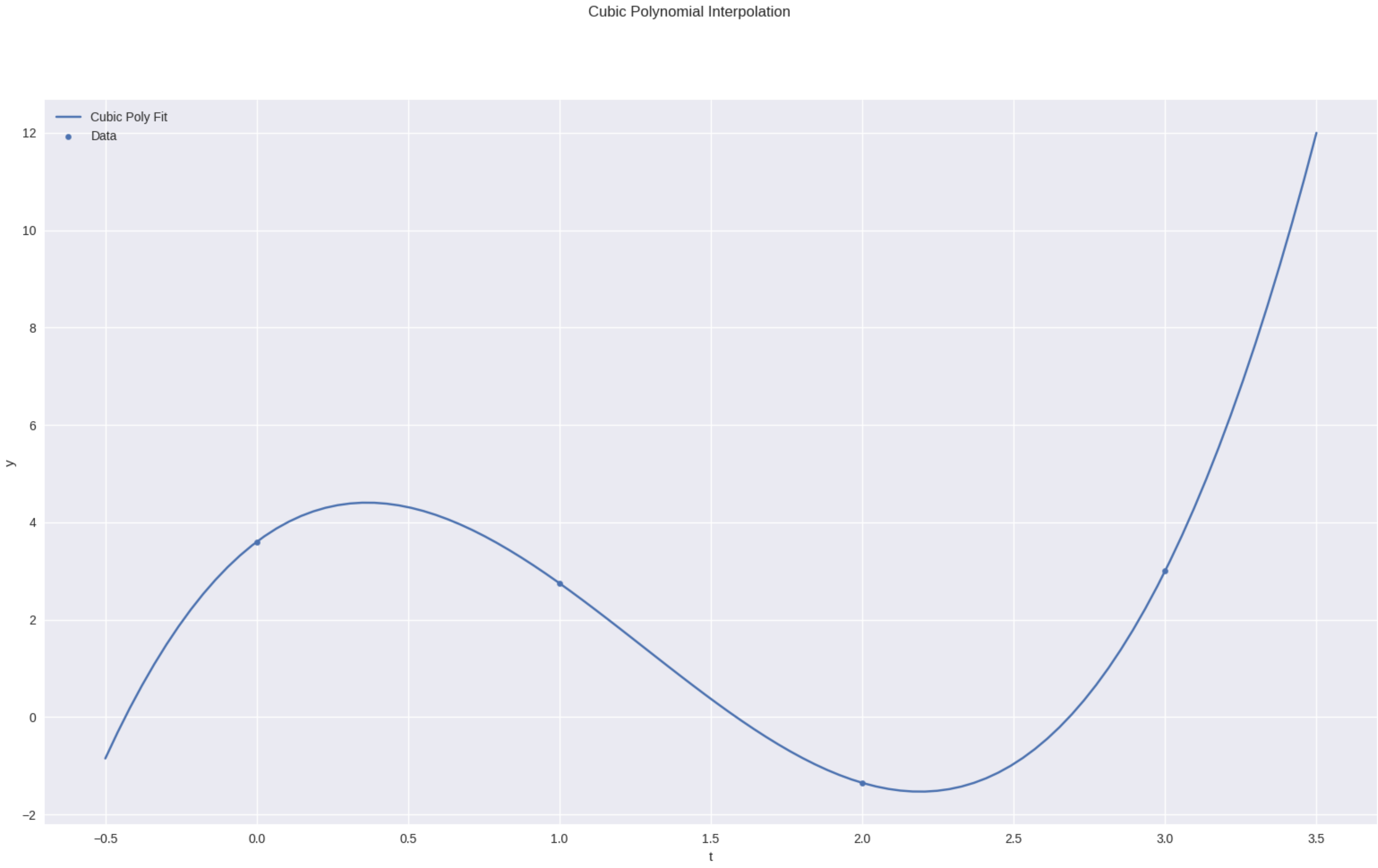
```
import sys
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt

mpl.style.use('seaborn')

t_vec = np.array([0, 1, 2, 3])
y_vec = np.array([3.6, 2.75, -1.35, 3.0])

def part_a(t_vec, y_vec):
    print("""
    Find cubic polynomial that interpolates data
    Formulate the problem as a system of equations, Ax = b
    => x = inv(A) * b
    Where b = y_vec^T and x = cubic polynomial coefficients
    x = [a, b, c, d] where our polynomial is at^3 + bt^2 + ct + d
    """)

    A = np.zeros(shape=(len(t_vec), len(t_vec)))
    b = np.zeros(shape=(len(t_vec), 1))
    for i in range(0, len(t_vec)):
        t = t_vec[i]
        y = y_vec[i]

        A[i,:] = [t**3, t**2, t**1, 1]
        b[i,:] = [y]

    print("A: ")
    print(A)

    print("b: ")
    print(b)

    x = np.linalg.inv(A) @ b
    print("x (a, b, c, d): ")
    print(x)

    # Plot result with data points overlaid
    fig = plt.figure()
    fig.suptitle("Cubic Polynomial Interpolation")
    ax = fig.add_subplot(111)
    ax.scatter(t_vec, y_vec, s = 20, label="Data")
    t_domain = np.linspace(min(t_vec) - 0.5, max(t_vec) + 0.5, 100)
    ax.plot(t_domain, [np.polyval(x, i) for i in t_domain], label="Cubic Poly
Fit")

    ax.set_xlabel("t")
    ax.set_ylabel("y")
    ax.legend()

    plt.show()

part_a(t_vec, y_vec)

def part_b(t_vec, y_vec):

    print("""
    Find a cubic spline that interpolates data
    Formulate the problem as a system of equations, Ax = b
    => x = inv(A) * b
    """)

    A = np.zeros(shape=(len(t_vec) * 3, len(t_vec) * 3))
```

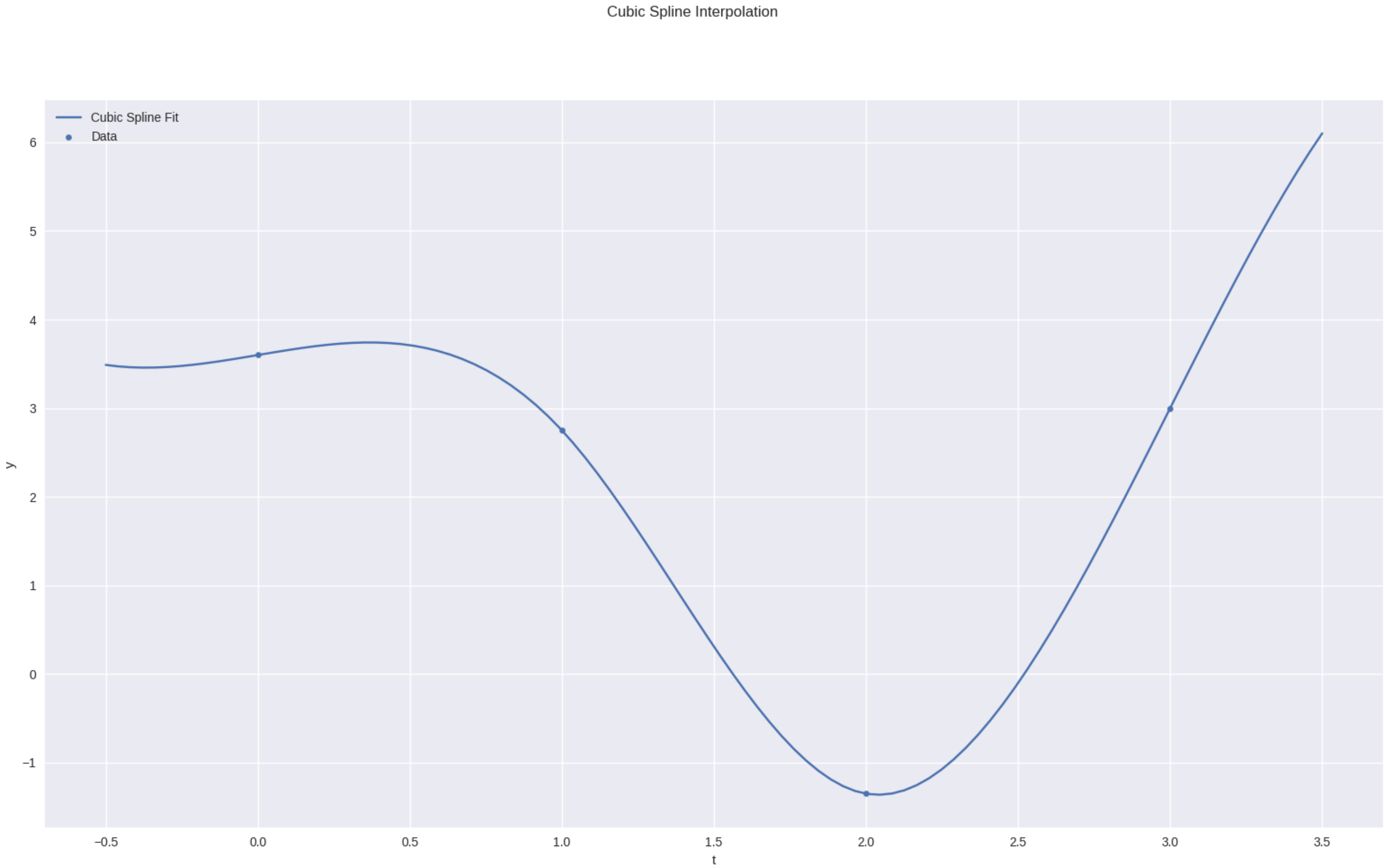
```
    b = np.zeros(shape=(len(t_vec) * 3, 1))

    A[0,:] = [t_vec[0]**3, t_vec[0]**2, t_vec[0]**1, 1, 0, 0, 0, 0, 0, 0, 0, 0]
    b[0,:] = [y_vec[0]]
    A[1,:] = [t_vec[1]**3, t_vec[1]**2, t_vec[1]**1, 1, 0, 0, 0, 0, 0, 0, 0, 0]
    b[1,:] = [y_vec[1]]
    A[2,:] = [0, 0, 0, 0, t_vec[1]**3, t_vec[1]**2, t_vec[1]**1, 1, 0, 0, 0, 0]
    b[2,:] = [y_vec[1]]
    A[3,:] = [0, 0, 0, 0, t_vec[2]**3, t_vec[2]**2, t_vec[2]**1, 1, 0, 0, 0, 0]
    b[3,:] = [y_vec[2]]
    A[4,:] = [0, 0, 0, 0, 0, 0, 0, 0, t_vec[2]**3, t_vec[2]**2, t_vec[2]**1, 1]
    b[4,:] = [y_vec[2]]
    A[5,:] = [0, 0, 0, 0, 0, 0, 0, 0, t_vec[3]**3, t_vec[3]**2, t_vec[3]**1, 1]
    b[5,:] = [y_vec[3]]

    # First and second derivs have to match at t1 & t2
    # First derivative relationship (3*a1*t^2 + 2*b1*t + c1 = 3*a2*t^2 + 2*b2*t
+ c2)
    # Second derivative relationship (6*a1*t + 2*b1 = 6*a2*t + 2*b2)
    A[6,:] = [3 * t_vec[1]**2, 2*t_vec[1], 1, 0, -3 * t_vec[1]**2, -2*t_vec[1],
-1, 0, 0, 0, 0, 0]
    b[6,:] = [0]
    A[7,:] = [6 * t_vec[1], 2, 0, 0, -6 * t_vec[1], -2, 0, 0, 0, 0, 0, 0]
    b[7,:] = [0]
    A[8,:] = [0, 0, 0, 0, 3 * t_vec[2]**2, 2*t_vec[2], 1, 0, -3 * t_vec[2]**2, -
2*t_vec[2], -1, 0]
    b[8,:] = [0]
    A[9,:] = [0, 0, 0, 0, 6 * t_vec[2], 2, 0, 0, -6 * t_vec[2], -2, 0, 0]
    b[9,:] = [0]

    # Second derivative (6*a*t + 2*b) at t0 & t3 = 0
    A[10,:] = [6 * t_vec[0], 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
    b[10,:] = [0]
    A[11,:] = [0, 0, 0, 0, 0, 0, 0, 0, 6 * t_vec[3], 2, 0, 0]
    b[11,:] = [0]


    print("A: ")
    print(A)

    print("b: ")
    print(b)

    x = np.linalg.inv(A) @ b
    print("x (a1, b1, c1, d1, a2, b2, c2, d2, a3, b3, c3, d3)")
    print(x)

    cube_poly_1 = x[0:4,:]
    cube_poly_2 = x[4:8,:]
    cube_poly_3 = x[8:12,:]

    # Plot result with data points overlaid
    fig = plt.figure()
    fig.suptitle("Cubic Spline Interpolation")
    ax = fig.add_subplot(111)
    ax.scatter(t_vec, y_vec, s = 20, label="Data")
    t_domain = np.linspace(min(t_vec) - 0.5, max(t_vec) + 0.5, 100)
    ax.plot(t_domain, [np.polyval(cube_poly_1, i) if i < t_vec[1] else
        (np.polyval(cube_poly_2, i) if i < t_vec[2] else np.polyval(cube_poly_3,
            i)) for i in t_domain], label="Cubic Spline Fit")

    ax.set_xlabel("t")
    ax.set_ylabel("y")
    ax.legend()
```

```
    plt.show()

part_b(t_vec, y_vec)
```

```
    Find cubic polynomial that interpolates data
    Formulate the problem as a system of equations, Ax = b
    => x = inv(A) * b
    Where b = y_vec^T and x = cubic polynomial coefficients
    x = [a, b, c, d] where our polynomial is at^3 + bt^2 + ct + d

A:
[[ 0.  0.  0.  1.]
 [ 1.  1.  1.  1.]
 [ 8.  4.  2.  1.]
 [27.  9.  3.  1.]]
b:
[[ 3.6 ]
 [ 2.75]
 [-1.35]
 [ 3.  ]]
x (a, b, c, d):
[[ 1.95 ]
 [-7.475]
 [ 4.675]
 [ 3.6  ]]

    Find a cubic spline that interpolates data
    Formulate the problem as a system of equations, Ax = b
    => x = inv(A) * b

A:
[[  0.   0.   0.   1.   0.   0.   0.   0.   0.   0.   0.   0.]
 [  1.   1.   1.   1.   0.   0.   0.   0.   0.   0.   0.   0.]
 [  0.   0.   0.   0.   1.   1.   1.   1.   0.   0.   0.   0.]
 [  0.   0.   0.   0.   8.   4.   2.   1.   0.   0.   0.   0.]
 [  0.   0.   0.   0.   0.   0.   0.   0.   8.   4.   2.   1.]
 [  0.   0.   0.   0.   0.   0.   0.   0.  27.   9.   3.   1.]
 [  3.   2.   1.   0.  -3.  -2.  -1.   0.   0.   0.   0.   0.]
 [  6.   2.   0.   0.  -6.  -2.   0.   0.   0.   0.   0.   0.]
 [  0.   0.   0.   0.  12.   4.   1.   0. -12.  -4.  -1.   0.]
 [  0.   0.   0.   0.  12.   2.   0.   0. -12.  -2.   0.   0.]
 [  0.   2.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.]
 [  0.   0.   0.   0.   0.   0.   0.   0.  18.   2.   0.   0.]]
b:
[[ 3.6 ]
 [ 2.75]
 [ 2.75]
 [-1.35]
 [-1.35]
 [ 3.  ]
 [ 0.  ]
 [ 0.  ]
 [ 0.  ]
 [ 0.  ]
 [ 0.  ]
 [ 0.  ]]
x (a1, b1, c1, d1, a2, b2, c2, d2, a3, b3, c3, d3)
[[ -1.43]
 [  0.  ]
 [  0.58]
 [  3.6 ]
 [  3.9 ]
 [-15.99]
 [ 16.57]
 [ -1.73]
 [ -2.47]
 [ 22.23]
```

```
[-59.87]
[ 49.23]]
```


Cubic Polynomial Interpolation
Cubic Poly Fit
Data
y
t
12
10
8
6
4
2
0
−2
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5


Cubic Spline Interpolation
Cubic Spline Fit
Data
y
t
6
5
4
3
2
1
0
−1
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5

```
import sys
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt

mpl.style.use('seaborn')

def ft(t):
    return 1/(1 + 25*(t**2))

def my_polyfit(P):
    t_vec = [-1 + 2*k/P for k in range(0, P + 1)]

    y_vec = np.zeros(len(t_vec))
    for i in range(0, len(t_vec)):
        y_vec[i] = ft(t_vec[i])

    print("Polyfitting to Order: " + str(P))
    A = np.zeros(shape=(len(t_vec), len(t_vec)))
    b = np.zeros(shape=(len(t_vec), 1))

    for i in range(0, len(t_vec)):
        t = t_vec[i]
        y = y_vec[i]

        A[i,:] = [t**i for i in reversed(range(0, P+1))]
        b[i,:] = [y]

    print("A: ")
    print(A)

    print("b: ")
    print(b)

    x = np.linalg.inv(A) @ b
    print("x (coefficients): ")
    print(x)
    return x

def plot_em_all(p_vec):
    t_vec = [-1, 1]

    # Plot result with data points overlaid
    for p in p_vec:
        fig = plt.figure()
        fig.suptitle("Polynomial Interpolation")
        ax = fig.add_subplot(111)
        t_domain = np.linspace(min(t_vec) - 0.001, max(t_vec) + 0.001, 1000)
        ax.plot(t_domain, [ft(i) for i in t_domain], label="f(t)",
color="tab:orange")

        x = my_polyfit(p)
        label_str = str(p) + "th Order Fit"
        ax.plot(t_domain, [np.polyval(x, i) for i in t_domain], label=label_str)

        ax.set_xlabel("t")
        ax.set_ylabel("y")
        ax.legend()

        plt.show()

    fig = plt.figure()
    fig.suptitle("Polynomial Interpolation")
    ax = fig.add_subplot(111)
```

```
    t_domain = np.linspace(min(t_vec) - 0.001, max(t_vec) + 0.001, 1000)
    ax.plot(t_domain, [ft(i) for i in t_domain], label="f(t)",
color="tab:orange")

    for p in p_vec:
        x = my_polyfit(p)
        label_str = str(p) + "th Order Fit"
        ax.plot(t_domain, [np.polyval(x, i) for i in t_domain], label=label_str)

    ax.set_xlabel("t")
    ax.set_ylabel("y")
    ax.legend()

    plt.show()

p_vec = [3, 5, 7, 9, 11, 15]

plot_em_all(p_vec)
```

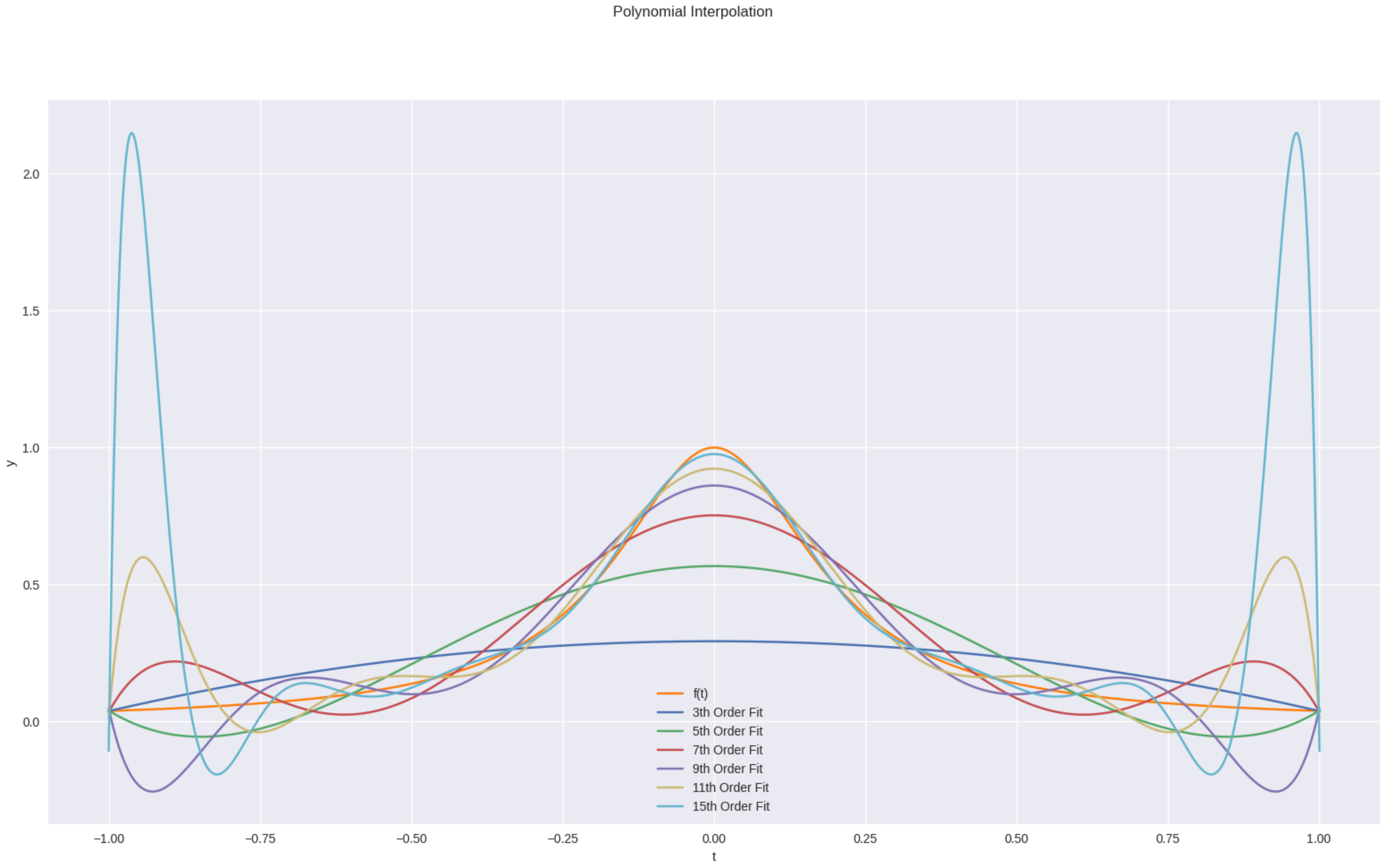
```
Polyfitting to Order: 3
A:
[[-1.          1.         -1.          1.        ]
 [-0.03703704  0.11111111 -0.33333333  1.        ]
 [ 0.03703704  0.11111111  0.33333333  1.        ]
 [ 1.          1.          1.          1.        ]]
b:
[[0.03846154]
 [0.26470588]
 [0.26470588]
 [0.03846154]]
x (coefficients):
[[-5.55111512e-17]
 [-2.54524887e-01]
 [ 5.55111512e-17]
 [ 2.92986425e-01]]
Polyfitting to Order: 5
A:
[[-1.000e+00  1.000e+00 -1.000e+00  1.000e+00 -1.000e+00  1.000e+00]
 [-7.776e-02  1.296e-01 -2.160e-01  3.600e-01 -6.000e-01  1.000e+00]
 [-3.200e-04  1.600e-03 -8.000e-03  4.000e-02 -2.000e-01  1.000e+00]
 [ 3.200e-04  1.600e-03  8.000e-03  4.000e-02  2.000e-01  1.000e+00]
 [ 7.776e-02  1.296e-01  2.160e-01  3.600e-01  6.000e-01  1.000e+00]
 [ 1.000e+00  1.000e+00  1.000e+00  1.000e+00  1.000e+00  1.000e+00]]
b:
[[0.03846154]
 [0.1       ]
 [0.5       ]
 [0.5       ]
 [0.1       ]
 [0.03846154]]
x (coefficients):
[[ 1.94289029e-15]
 [ 1.20192308e+00]
 [-1.60982339e-15]
 [-1.73076923e+00]
 [ 6.45317133e-16]
 [ 5.67307692e-01]]
Polyfitting to Order: 7
A:
[[-1.00000000e+00  1.00000000e+00 -1.00000000e+00  1.00000000e+00
  -1.00000000e+00  1.00000000e+00 -1.00000000e+00  1.00000000e+00]
 [-9.48645062e-02  1.32810309e-01 -1.85934432e-01  2.60308205e-01
  -3.64431487e-01  5.10204082e-01 -7.14285714e-01  1.00000000e+00]
 [-2.65559904e-03  6.19639776e-03 -1.44582614e-02  3.37359434e-02
  -7.87172012e-02  1.83673469e-01 -4.28571429e-01  1.00000000e+00]
 [-1.21426568e-06  8.49985975e-06 -5.94990183e-05  4.16493128e-04
  -2.91545190e-03  2.04081633e-02 -1.42857143e-01  1.00000000e+00]
 [ 1.21426568e-06  8.49985975e-06  5.94990183e-05  4.16493128e-04
   2.91545190e-03  2.04081633e-02  1.42857143e-01  1.00000000e+00]
 [ 2.65559904e-03  6.19639776e-03  1.44582614e-02  3.37359434e-02
   7.87172012e-02  1.83673469e-01  4.28571429e-01  1.00000000e+00]
 [ 9.48645062e-02  1.32810309e-01  1.85934432e-01  2.60308205e-01
   3.64431487e-01  5.10204082e-01  7.14285714e-01  1.00000000e+00]
 [ 1.00000000e+00  1.00000000e+00  1.00000000e+00  1.00000000e+00
   1.00000000e+00  1.00000000e+00  1.00000000e+00  1.00000000e+00]]
b:
[[0.03846154]
 [0.0727003 ]
 [0.17883212]
 [0.66216216]
 [0.66216216]
 [0.17883212]
 [0.0727003 ]
```

```
 [0.03846154]]
x (coefficients):
[[-3.55271368e-14]
 [-5.17359870e+00]
 [ 4.26325641e-14]
 [ 9.07597030e+00]
 [-1.06581410e-14]
 [-4.61655145e+00]
 [ 1.33226763e-15]
 [ 7.52641393e-01]]
Polyfitting to Order: 9
A:
[[-1.00000000e+00  1.00000000e+00 -1.00000000e+00  1.00000000e+00
  -1.00000000e+00  1.00000000e+00 -1.00000000e+00  1.00000000e+00
  -1.00000000e+00  1.00000000e+00]
 [-1.04159713e-01  1.33919631e-01 -1.72182383e-01  2.21377350e-01
  -2.84628021e-01  3.65950312e-01 -4.70507545e-01  6.04938272e-01
  -7.77777778e-01  1.00000000e+00]
 [-5.04135702e-03  9.07444263e-03 -1.63339967e-02  2.94011941e-02
  -5.29221494e-02  9.52598689e-02 -1.71467764e-01  3.08641975e-01
  -5.55555556e-01  1.00000000e+00]
 [-5.08052634e-05  1.52415790e-04 -4.57247371e-04  1.37174211e-03
  -4.11522634e-03  1.23456790e-02 -3.70370370e-02  1.11111111e-01
  -3.33333333e-01  1.00000000e+00]
 [-2.58117479e-09  2.32305731e-08 -2.09075158e-07  1.88167642e-06
  -1.69350878e-05  1.52415790e-04 -1.37174211e-03  1.23456790e-02
  -1.11111111e-01  1.00000000e+00]
 [ 2.58117479e-09  2.32305731e-08  2.09075158e-07  1.88167642e-06
   1.69350878e-05  1.52415790e-04  1.37174211e-03  1.23456790e-02
   1.11111111e-01  1.00000000e+00]
 [ 5.08052634e-05  1.52415790e-04  4.57247371e-04  1.37174211e-03
   4.11522634e-03  1.23456790e-02  3.70370370e-02  1.11111111e-01
   3.33333333e-01  1.00000000e+00]
 [ 5.04135702e-03  9.07444263e-03  1.63339967e-02  2.94011941e-02
   5.29221494e-02  9.52598689e-02  1.71467764e-01  3.08641975e-01
   5.55555556e-01  1.00000000e+00]
 [ 1.04159713e-01  1.33919631e-01  1.72182383e-01  2.21377350e-01
   2.84628021e-01  3.65950312e-01  4.70507545e-01  6.04938272e-01
   7.77777778e-01  1.00000000e+00]
 [ 1.00000000e+00  1.00000000e+00  1.00000000e+00  1.00000000e+00
   1.00000000e+00  1.00000000e+00  1.00000000e+00  1.00000000e+00
   1.00000000e+00  1.00000000e+00]]
b:
[[0.03846154]
 [0.06202144]
 [0.11473088]
 [0.26470588]
 [0.76415094]
 [0.76415094]
 [0.26470588]
 [0.11473088]
 [0.06202144]
 [0.03846154]]
x (coefficients):
[[ 1.14575016e-13]
 [ 2.16247748e+01]
 [-2.13162821e-13]
 [-4.49154581e+01]
 [ 1.01030295e-13]
 [ 3.07285300e+01]
 [-1.15948917e-14]
 [-8.26092333e+00]
 [ 2.10335221e-17]
 [ 8.61538152e-01]]
```

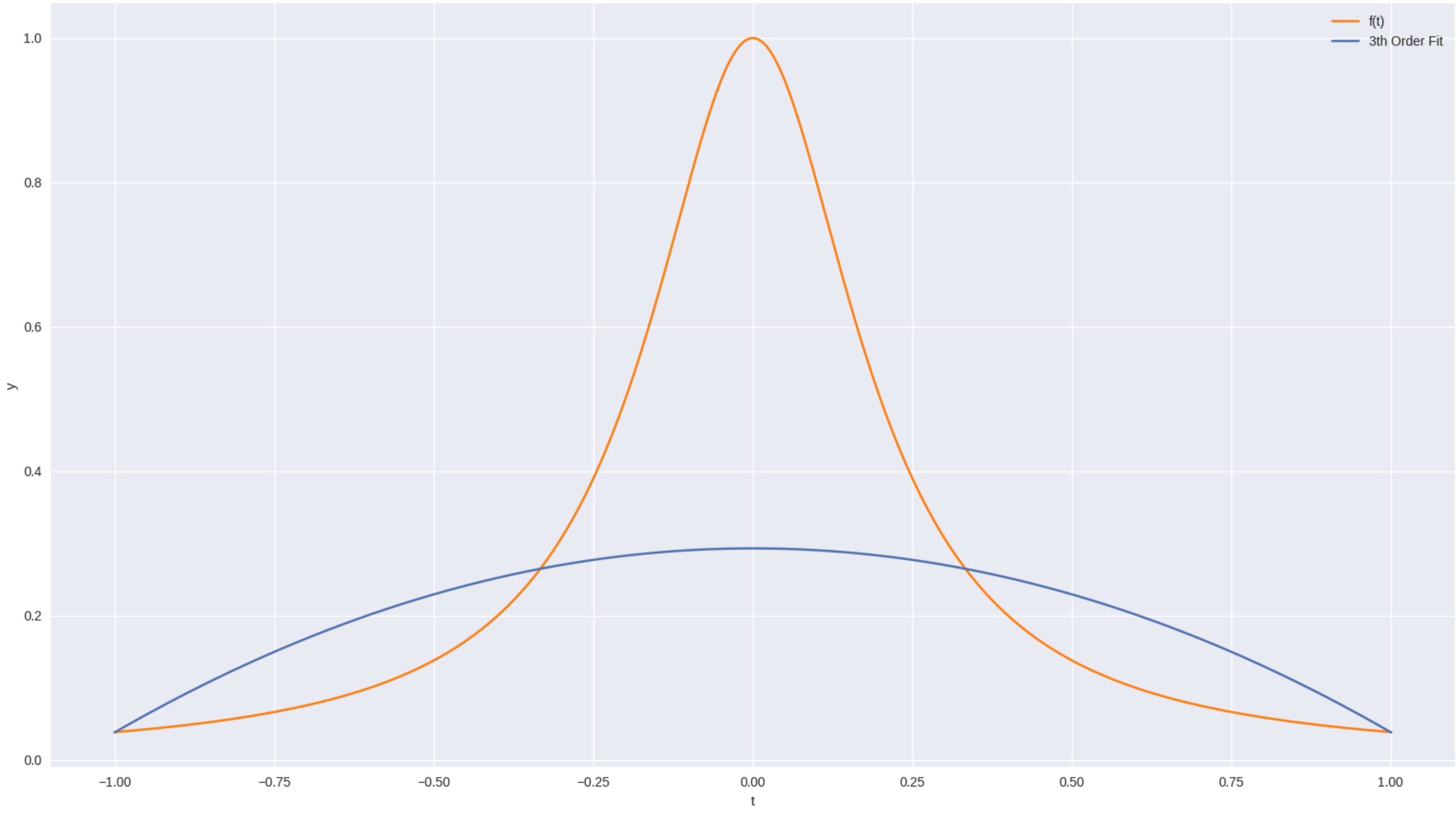
```
Polyfitting to Order: 11
A:
[[-1.00000000e+00  1.00000000e+00 -1.00000000e+00  1.00000000e+00
  -1.00000000e+00  1.00000000e+00 -1.00000000e+00  1.00000000e+00
  -1.00000000e+00  1.00000000e+00 -1.00000000e+00  1.00000000e+00]
 [-1.09988700e-01  1.34430633e-01 -1.64304107e-01  2.00816130e-01
  -2.45441937e-01  2.99984590e-01 -3.66647832e-01  4.48125128e-01
  -5.47708490e-01  6.69421488e-01 -8.18181818e-01  1.00000000e+00]
 [-6.93040961e-03  1.08906437e-02 -1.71138686e-02  2.68932221e-02
  -4.22607776e-02  6.64097934e-02 -1.04358247e-01  1.63991531e-01
  -2.57700977e-01  4.04958678e-01 -6.36363636e-01  1.00000000e+00]
 [-1.71139599e-04  3.76507119e-04 -8.28315661e-04  1.82229445e-03
  -4.00904780e-03  8.81990516e-03 -1.94037913e-02  4.26883410e-02
  -9.39143501e-02  2.06611570e-01 -4.54545455e-01  1.00000000e+00]
 [-6.20889428e-07  2.27659457e-06 -8.34751342e-06  3.06075492e-05
  -1.12227680e-04  4.11501495e-04 -1.50883882e-03  5.53240899e-03
  -2.02854996e-02  7.43801653e-02 -2.72727273e-01  1.00000000e+00]
 [-3.50493899e-12  3.85543289e-11 -4.24097618e-10  4.66507380e-09
  -5.13158118e-08  5.64473930e-07 -6.20921323e-06  6.83013455e-05
  -7.51314801e-04  8.26446281e-03 -9.09090909e-02  1.00000000e+00]
 [ 3.50493899e-12  3.85543289e-11  4.24097618e-10  4.66507380e-09
   5.13158118e-08  5.64473930e-07  6.20921323e-06  6.83013455e-05
   7.51314801e-04  8.26446281e-03  9.09090909e-02  1.00000000e+00]
 [ 6.20889428e-07  2.27659457e-06  8.34751342e-06  3.06075492e-05
   1.12227680e-04  4.11501495e-04  1.50883882e-03  5.53240899e-03
   2.02854996e-02  7.43801653e-02  2.72727273e-01  1.00000000e+00]
 [ 1.71139599e-04  3.76507119e-04  8.28315661e-04  1.82229445e-03
   4.00904780e-03  8.81990516e-03  1.94037913e-02  4.26883410e-02
   9.39143501e-02  2.06611570e-01  4.54545455e-01  1.00000000e+00]
 [ 6.93040961e-03  1.08906437e-02  1.71138686e-02  2.68932221e-02
   4.22607776e-02  6.64097934e-02  1.04358247e-01  1.63991531e-01
   2.57700977e-01  4.04958678e-01  6.36363636e-01  1.00000000e+00]
 [ 1.09988700e-01  1.34430633e-01  1.64304107e-01  2.00816130e-01
   2.45441937e-01  2.99984590e-01  3.66647832e-01  4.48125128e-01
   5.47708490e-01  6.69421488e-01  8.18181818e-01  1.00000000e+00]
 [ 1.00000000e+00  1.00000000e+00  1.00000000e+00  1.00000000e+00
   1.00000000e+00  1.00000000e+00  1.00000000e+00  1.00000000e+00
   1.00000000e+00  1.00000000e+00  1.00000000e+00  1.00000000e+00]]
b:
[[0.03846154]
 [0.05638397]
 [0.08989599]
 [0.16219839]
 [0.34971098]
 [0.82876712]
 [0.82876712]
 [0.34971098]
 [0.16219839]
 [0.08989599]
 [0.05638397]
 [0.03846154]]
x (coefficients):
[[ 2.27373675e-13]
 [-8.94975172e+01]
 [-9.09494702e-13]
 [ 2.15119487e+02]
 [ 0.00000000e+00]
 [-1.84305014e+02]
 [ 0.00000000e+00]
 [ 6.97605574e+01]
 [-2.84217094e-14]
 [-1.19620163e+01]
 [ 8.88178420e-16]
 [ 9.22965058e-01]]
```

```
Polyfitting to Order: 15
A:
[[-1.00000000e+00  1.00000000e+00 -1.00000000e+00  1.00000000e+00
  -1.00000000e+00  1.00000000e+00 -1.00000000e+00  1.00000000e+00
  -1.00000000e+00  1.00000000e+00 -1.00000000e+00  1.00000000e+00
  -1.00000000e+00  1.00000000e+00 -1.00000000e+00  1.00000000e+00]
 [-1.16891087e-01  1.34874332e-01 -1.55624229e-01  1.79566418e-01
  -2.07192021e-01  2.39067716e-01 -2.75847365e-01  3.18285421e-01
  -3.67252409e-01  4.23752779e-01 -4.88945514e-01  5.64167901e-01
  -6.50962963e-01  7.51111111e-01 -8.66666667e-01  1.00000000e+00]
 [-9.53940729e-03  1.30082827e-02 -1.77385673e-02  2.41889554e-02
  -3.29849391e-02  4.49794625e-02 -6.13356306e-02  8.36394963e-02
  -1.14053859e-01  1.55527989e-01 -2.12083621e-01  2.89204938e-01
  -3.94370370e-01  5.37777778e-01 -7.33333333e-01  1.00000000e+00]
 [-4.70184985e-04  7.83641641e-04 -1.30606940e-03  2.17678234e-03
  -3.62797056e-03  6.04661760e-03 -1.00776960e-02  1.67961600e-02
  -2.79936000e-02  4.66560000e-02 -7.77600000e-02  1.29600000e-01
  -2.16000000e-01  3.60000000e-01 -6.00000000e-01  1.00000000e+00]
 [-1.08418081e-05  2.32324458e-05 -4.97838125e-05  1.06679598e-04
  -2.28599139e-04  4.89855298e-04 -1.04968992e-03  2.24933555e-03
  -4.82000476e-03  1.03285816e-02 -2.21326749e-02  4.74271605e-02
  -1.01629630e-01  2.17777778e-01 -4.66666667e-01  1.00000000e+00]
 [-6.96917194e-08  2.09075158e-07 -6.27225474e-07  1.88167642e-06
  -5.64502927e-06  1.69350878e-05 -5.08052634e-05  1.52415790e-04
  -4.57247371e-04  1.37174211e-03 -4.11522634e-03  1.23456790e-02
  -3.70370370e-02  1.11111111e-01 -3.33333333e-01  1.00000000e+00]
 [-3.27680000e-11  1.63840000e-10 -8.19200000e-10  4.09600000e-09
  -2.04800000e-08  1.02400000e-07 -5.12000000e-07  2.56000000e-06
  -1.28000000e-05  6.40000000e-05 -3.20000000e-04  1.60000000e-03
  -8.00000000e-03  4.00000000e-02 -2.00000000e-01  1.00000000e+00]
 [-2.28365826e-18  3.42548739e-17 -5.13823109e-16  7.70734663e-15
  -1.15610199e-13  1.73415299e-12 -2.60122949e-11  3.90184423e-10
  -5.85276635e-09  8.77914952e-08 -1.31687243e-06  1.97530864e-05
  -2.96296296e-04  4.44444444e-03 -6.66666667e-02  1.00000000e+00]
 [ 2.28365826e-18  3.42548739e-17  5.13823109e-16  7.70734663e-15
   1.15610199e-13  1.73415299e-12  2.60122949e-11  3.90184423e-10
   5.85276635e-09  8.77914952e-08  1.31687243e-06  1.97530864e-05
   2.96296296e-04  4.44444444e-03  6.66666667e-02  1.00000000e+00]
 [ 3.27680000e-11  1.63840000e-10  8.19200000e-10  4.09600000e-09
   2.04800000e-08  1.02400000e-07  5.12000000e-07  2.56000000e-06
   1.28000000e-05  6.40000000e-05  3.20000000e-04  1.60000000e-03
   8.00000000e-03  4.00000000e-02  2.00000000e-01  1.00000000e+00]
 [ 6.96917194e-08  2.09075158e-07  6.27225474e-07  1.88167642e-06
   5.64502927e-06  1.69350878e-05  5.08052634e-05  1.52415790e-04
   4.57247371e-04  1.37174211e-03  4.11522634e-03  1.23456790e-02
   3.70370370e-02  1.11111111e-01  3.33333333e-01  1.00000000e+00]
 [ 1.08418081e-05  2.32324458e-05  4.97838125e-05  1.06679598e-04
   2.28599139e-04  4.89855298e-04  1.04968992e-03  2.24933555e-03
   4.82000476e-03  1.03285816e-02  2.21326749e-02  4.74271605e-02
   1.01629630e-01  2.17777778e-01  4.66666667e-01  1.00000000e+00]
 [ 4.70184985e-04  7.83641641e-04  1.30606940e-03  2.17678234e-03
   3.62797056e-03  6.04661760e-03  1.00776960e-02  1.67961600e-02
   2.79936000e-02  4.66560000e-02  7.77600000e-02  1.29600000e-01
   2.16000000e-01  3.60000000e-01  6.00000000e-01  1.00000000e+00]
 [ 9.53940729e-03  1.30082827e-02  1.77385673e-02  2.41889554e-02
   3.29849391e-02  4.49794625e-02  6.13356306e-02  8.36394963e-02
   1.14053859e-01  1.55527989e-01  2.12083621e-01  2.89204938e-01
   3.94370370e-01  5.37777778e-01  7.33333333e-01  1.00000000e+00]
 [ 1.16891087e-01  1.34874332e-01  1.55624229e-01  1.79566418e-01
   2.07192021e-01  2.39067716e-01  2.75847365e-01  3.18285421e-01
   3.67252409e-01  4.23752779e-01  4.88945514e-01  5.64167901e-01
   6.50962963e-01  7.51111111e-01  8.66666667e-01  1.00000000e+00]
 [ 1.00000000e+00  1.00000000e+00  1.00000000e+00  1.00000000e+00
   1.00000000e+00  1.00000000e+00  1.00000000e+00  1.00000000e+00
```

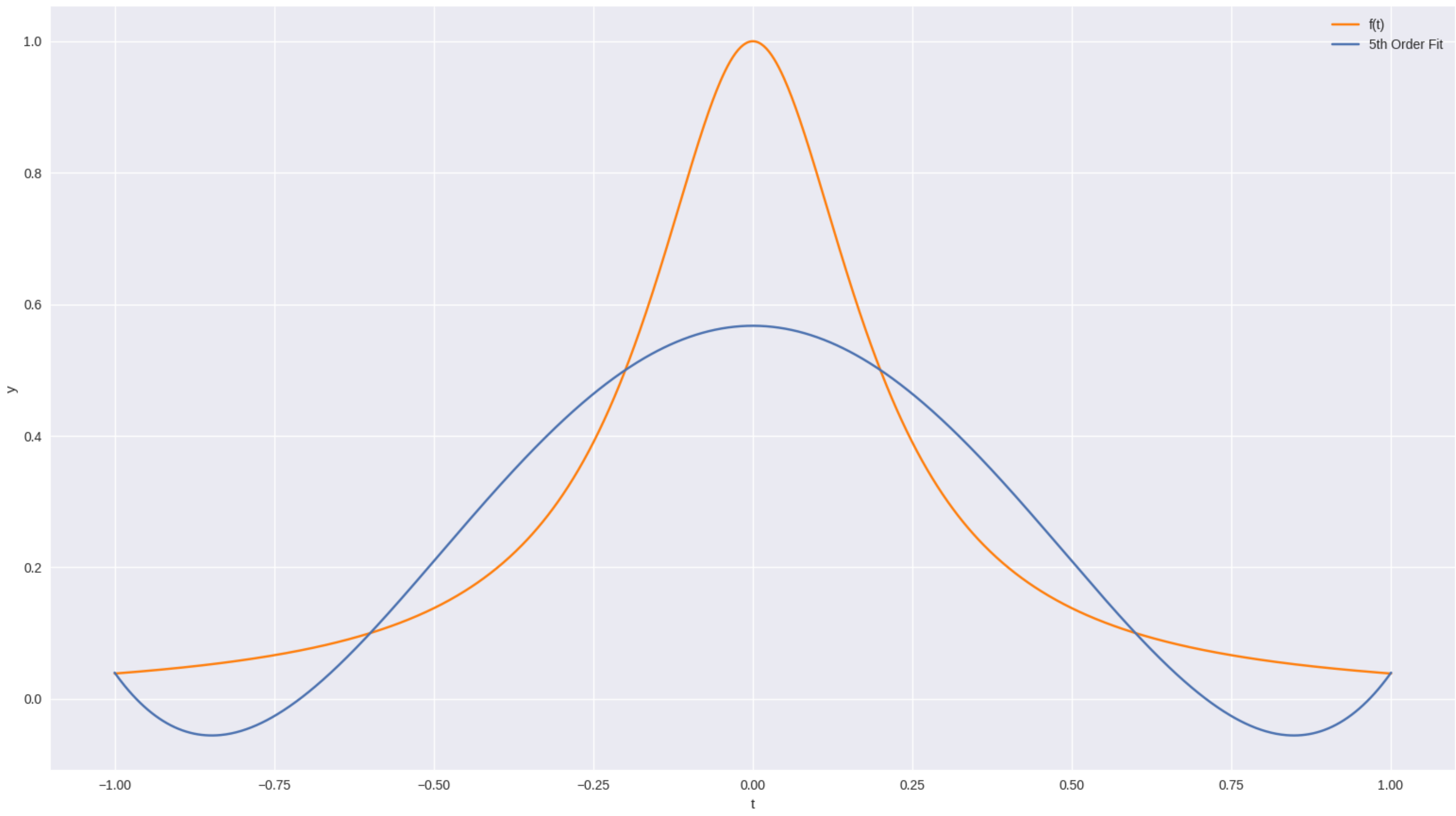
```
   1.00000000e+00  1.00000000e+00  1.00000000e+00  1.00000000e+00
   1.00000000e+00  1.00000000e+00  1.00000000e+00  1.00000000e+00]]
b:
[[0.03846154]
 [0.0505618 ]
 [0.06923077]
 [0.1       ]
 [0.15517241]
 [0.26470588]
 [0.5       ]
 [0.9       ]
 [0.9       ]
 [0.5       ]
 [0.26470588]
 [0.15517241]
 [0.1       ]
 [0.06923077]
 [0.0505618 ]
 [0.03846154]]
x (coefficients):
[[-1.60071068e-10]
 [-1.51886439e+03]
 [ 5.82076609e-10]
 [ 4.65110028e+03]
 [-5.23868948e-10]
 [-5.57000948e+03]
 [ 2.03726813e-10]
 [ 3.34768105e+03]
 [-2.18278728e-11]
 [-1.08300943e+03]
 [-1.81898940e-12]
 [ 1.90143756e+02]
 [ 0.00000000e+00]
 [-1.79795745e+01]
 [ 0.00000000e+00]
 [ 9.76247076e-01]]
```

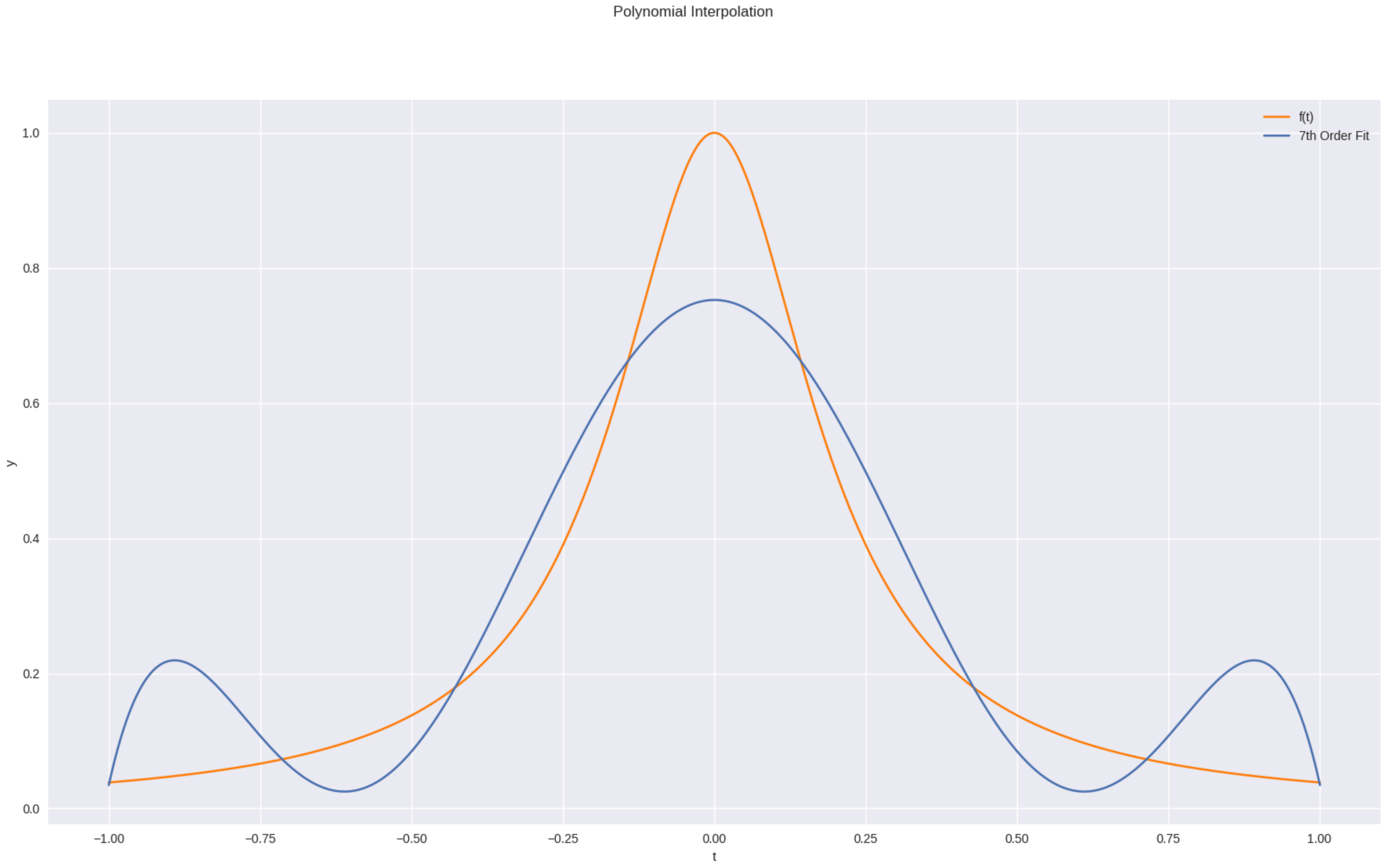

Polynomial Interpolation
2.0
1.5
1.0
0.5
0.0
y
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
t
f(t)
3th Order Fit
5th Order Fit
7th Order Fit
9th Order Fit
11th Order Fit
15th Order Fit

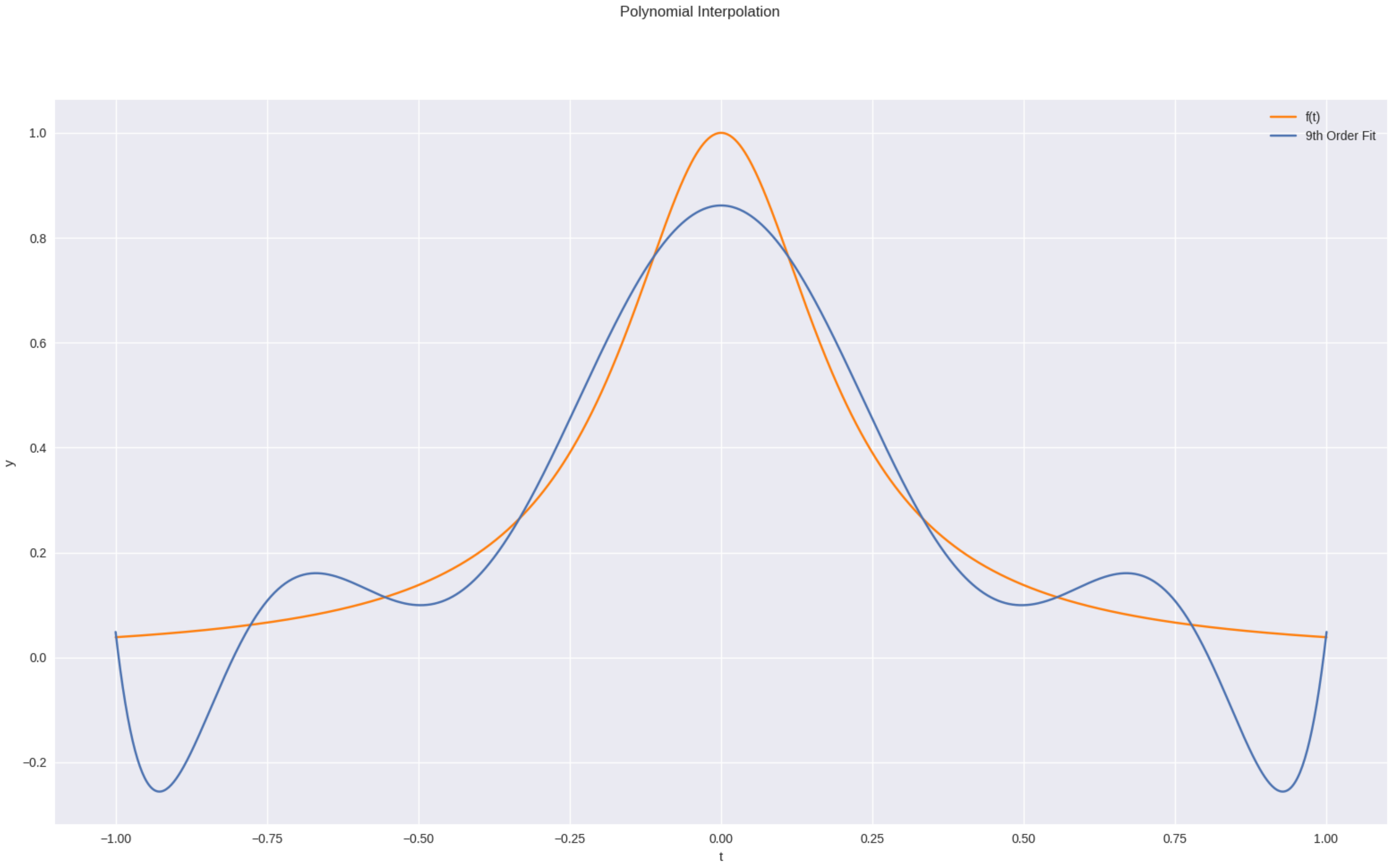
Polynomial Interpolation



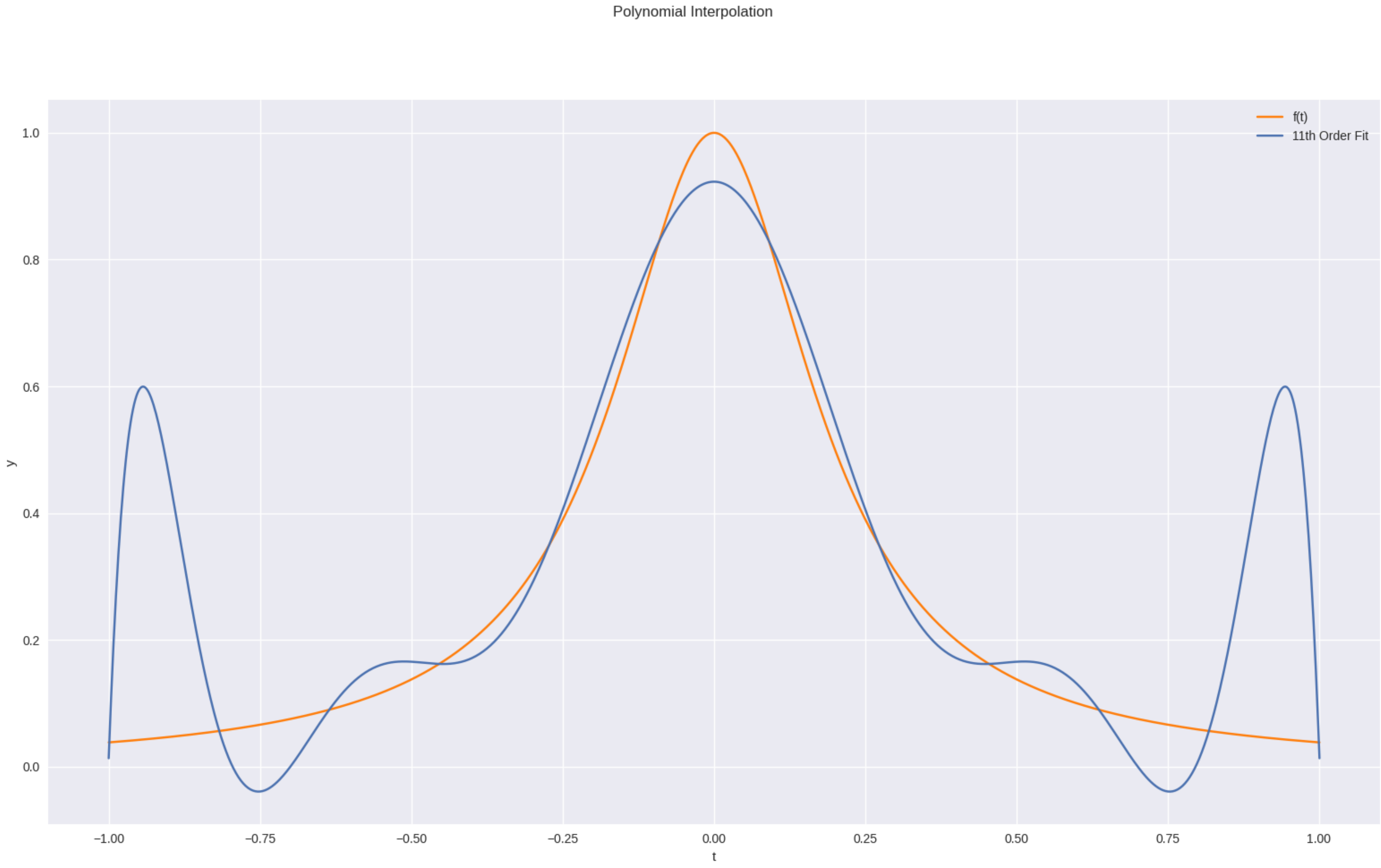
Polynomial Interpolation

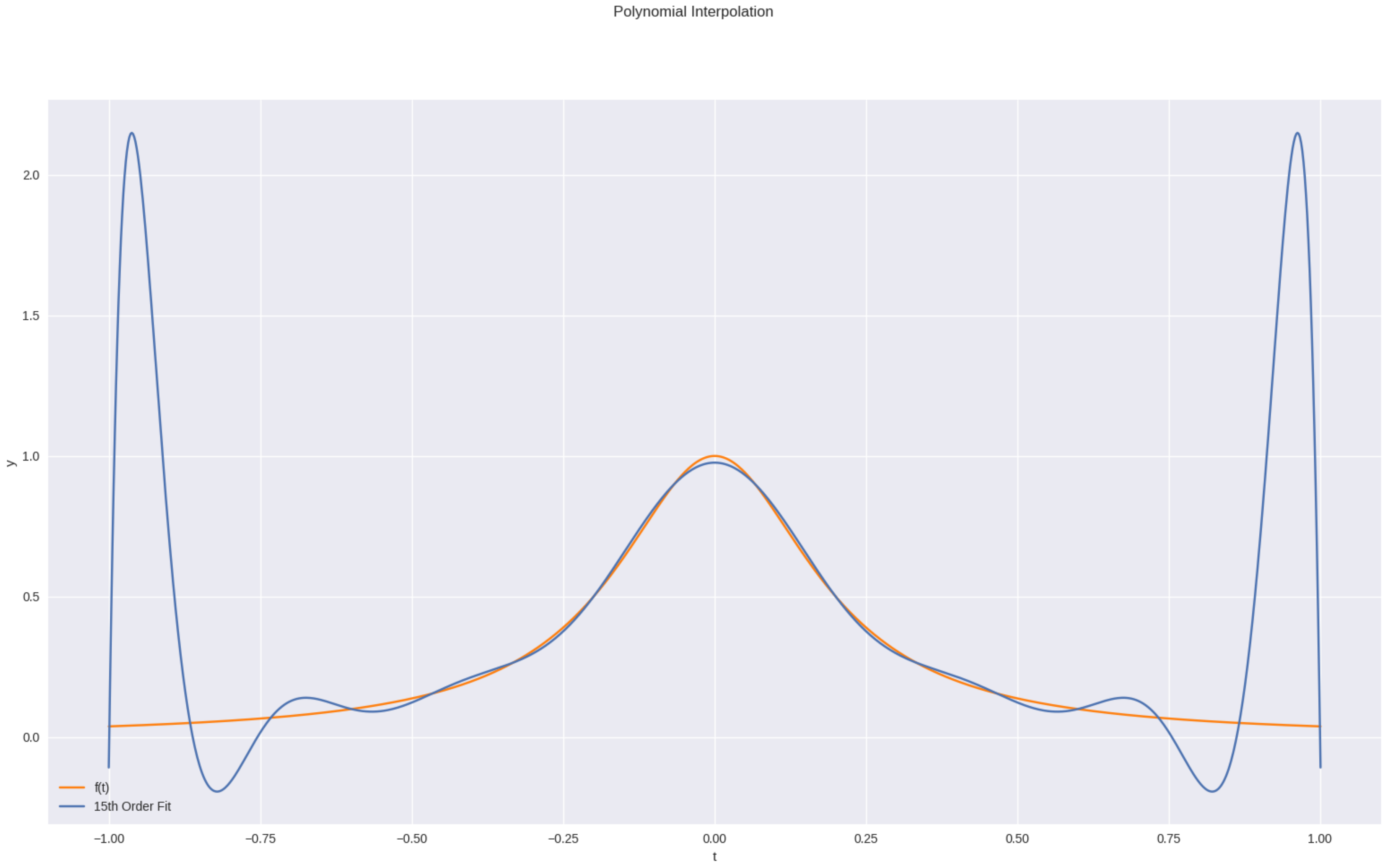



Polynomial Interpolation
f(t)
7th Order Fit
y
t
1.0
0.8
0.6
0.4
0.2
0.0
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00



Polynomial Interpolation
f(t)
9th Order Fit
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
y
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
t


Polynomial Interpolation
f(t)
11th Order Fit
1.0
0.8
0.6
0.4
0.2
0.0
y
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
t


Polynomial Interpolation
y
t
2.0
1.5
1.0
0.5
0.0
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
f(t)
15th Order Fit

```
import sys
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt

mpl.style.use('seaborn')

def b2(t):
    b2_val = 0
    if t >= -3/2 and t <= -1/2:
        b2_val = ((t + 3/2)**2)/2
    if t >= -1/2 and t <= 1/2:
        b2_val = -(t**2) + 3/4
    if t >= 1/2 and t <= 3/2:
        b2_val = ((t - 3/2)**2)/2
    if abs(t) >= 3/2:
        b2_val = 0
    return b2_val

# Part a
def piecepoly2(t, alpha):
    y_vec = np.zeros(len(t))
    for idx in range(0, len(t)):
        y_vec[idx] = sum(alpha[i] * b2(t[idx] - i) for i in range(0,
len(alpha)))
    return y_vec

def part_a():
    alpha = [3, 2, -1, 4, -1]
    t_vec = [-2, 6]

    # Plot result with data points overlaid
    fig = plt.figure()
    fig.suptitle("piecepoly2")
    ax = fig.add_subplot(111)
    t_domain = np.linspace(min(t_vec) - 0.5, max(t_vec) + 0.5, 100)
    ax.plot(t_domain, piecepoly2(t_domain, alpha), label="piecepoly2")

    ax.set_xlabel("t")
    ax.set_ylabel("y")
    ax.legend()

    plt.show()

part_a()

# Part b
def part_b():
    t_vec = np.array([0, 1, 2, 3, 4])
    y_vec = np.array([1, 2, -4, -5, -2])

    A = np.zeros(shape=(len(t_vec), len(t_vec)))
    b = np.zeros(shape=(len(t_vec), 1))
    for i in range(0, len(t_vec)):
        t = t_vec[i]
        y = y_vec[i]

        A[i,:] = [b2(t - 0), b2(t - 1), b2(t - 2), b2(t - 3), b2(t - 4)]
        b[i,:] = [y]

    print("A: ")
    print(A)

    print("b: ")
```

```
    print(b)

    x = np.linalg.inv(A) @ b
    print("x (alpha0, alpha1, alpha2, alpha3, alpha4): ")
    print(x)

print("Part b")
part_b()

# For Parts (c)-(f) see handwritten submission
```

Part b
A:

```
[[0.75  0.125 0.    0.    0.   ]
 [0.125 0.75  0.125 0.    0.   ]
 [0.    0.125 0.75  0.125 0.   ]
 [0.    0.    0.125 0.75  0.125]
 [0.    0.    0.    0.125 0.75 ]]
```

b:

```
[[ 1.]
 [ 2.]
 [-4.]
 [-5.]
 [-2.]]
```

x (alpha0, alpha1, alpha2, alpha3, alpha4):

```
[[ 0.77229437]
 [ 3.36623377]
 [-4.96969697]
 [-5.54805195]
 [-1.74199134]]
```

5c) Suppose $f(t)$ is now a superposition of N B-splines:

$$f(t) = \sum_{n=0}^{N-1} a_n b_2(t-n)$$

Describe how to construct the $N \times N$ matrix that maps the coefficients $\alpha$ to the N samples $f(0), \ldots, f(N-1)$. That is find A such that

$$\begin{bmatrix} f(0) \\ \vdots \\ f(N-1) \end{bmatrix} = A \begin{bmatrix} \alpha_0 \\ \vdots \\ \alpha_{N-1} \end{bmatrix}$$

The entries of A will correspond directly to the output of function $b_2(t-n)$ as all $\alpha_n$ terms will be multiplied with a corresponding A row vector to obtain $f(0), \ldots, f(N-1)$.

The inputs to $b_2$ will only be integers and thus $b_2(t)$ can only take on 3 different values.

When $t=-1$

$b_2(-1) = (-1+\frac{3}{2})^2/2 = \frac{1}{8}$

When $t=0$

$b_2(0) = -(0)^2 + \frac{3}{4} = \frac{3}{4}$

When $t=1$

$b_2(1) = (1-\frac{3}{2})^2/2 = \frac{1}{8}$

Elsewhere

$b_2 = 0$

Then

$$f(0) = a_0 \underbrace{b_2(0-0)}_{\frac{3}{4}} + a_1 \underbrace{b_2(0-1)}_{\frac{1}{8}} + \underbrace{\cdots}_{0}$$

$$f(1) = a_0 \underbrace{b_2(1-0)}_{\frac{1}{8}} + a_1 \underbrace{b_2(1-1)}_{\frac{3}{4}} + a_2 \underbrace{b_2(1-2)}_{\frac{1}{8}} + \underbrace{\cdots}_{0}$$

and for every subsequent row this sequence of $0, \ldots, \frac{1}{8}, \frac{3}{4}, \frac{1}{8}, \ldots, 0$ will be shifted by 1.

Thus $A = \begin{bmatrix} \frac{3}{4} & \frac{1}{8} & & & 0 \\ \frac{1}{8} & \frac{3}{4} & \frac{1}{8} & & \\ & \ddots & \ddots & \ddots & \\ & & & & \frac{1}{8} \\ 0 & & & \frac{1}{8} & \frac{3}{4} \end{bmatrix}$

5d) $A = \begin{bmatrix} \frac{3}{4} & \frac{1}{8} & 0 & & & \\ \frac{1}{8} & \frac{3}{4} & \frac{1}{8} & & & \\ 0 & \frac{1}{8} & \frac{3}{4} & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & \frac{1}{8} & \frac{3}{4} & \frac{1}{8} \\ & & & 0 & \frac{1}{8} & \frac{3}{4} \end{bmatrix} = \underset{a}{\underbrace{\frac{3}{4}}} I + \underbrace{\begin{bmatrix} 0 & \frac{1}{8} & & & \\ \frac{1}{8} & 0 & \frac{1}{8} & & \\ & \frac{1}{8} & 0 & \ddots & \\ & & \ddots & \ddots & \frac{1}{8} \\ & & & \frac{1}{8} & 0 \end{bmatrix}}_{G}$

$$\|Ax\| = \|(aI+G)x\| = \|ax+Gx\| \geq a\|x\| - \|Gx\| \geq 0$$

$$a\|x\| \geq \|Gx\|$$

$$(a\|x\|)^2 \geq (\|Gx\|)^2$$

$$\sum_{i=1}^{N} \left(\tfrac{3}{4}x_i\right)^2 \geq \left(\tfrac{1}{8}x_2\right)^2 + \left(\tfrac{1}{8}x_{N-1}\right)^2 + \sum_{2}^{N-1}\left(\tfrac{1}{8}(x_{i-1}+x_{i+1})^2\right)$$

$$\frac{9}{16}\sum_{i=1}^{N} x_i^2 \geq \frac{1}{64}x_2^2 + \frac{1}{64}x_{N-1}^2 + \sum_{i=2}^{N-1}\frac{1}{64}\left(x_{i-1}^2 + x_{i+1}^2 + 2x_{i-1}\cdot x_{i+1}\right)$$

$$\Rightarrow \frac{36}{64}\sum_{i=1}^{N} x_i^2 - \frac{1}{64}x_2^2 - \frac{1}{64}x_{N-1}^2 - \sum_{i=2}^{N-1}\frac{1}{64}\left(x_{i-1}^2 + x_{i+1}^2 + 2x_{i-1}\cdot x_{i+1}\right) \geq 0$$

$$\Rightarrow 36\sum_{i=1}^{N} x_i^2 - x_2^2 - x_{N-1}^2 - \sum_{2}^{N-1}\left(x_{i-1}^2 + x_{i+1}^2 + 2x_{i-1}x_{i+1}\right) \geq 0$$

$$\Rightarrow 32\sum_{i=1}^{N} x_i^2 + 4\sum_{i=1}^{N} x_i^2 - x_2^2 - x_{N-1}^2 - \sum_{i=2}^{N-1}\left(x_{i-1}^2 + x_{i+1}^2 + 2x_{i-1}x_{i+1}\right) \geq 0$$

$$\Rightarrow 32\sum_{i=1}^{N} x_i^2 + 4x_1^2 + 3x_2^2 + 3x_{N-1}^2 + 4x_N^2 + 4\sum_{i=3}^{N-2} x_i^2 - \sum_{i=2}^{N-1}\left(x_{i-1}^2 + x_{i+1}^2 + 2x_{i-1}x_{i+1}\right) \geq 0$$

$$\Rightarrow 32\sum_{i=1}^{N} x_i^2 + 4x_1^2 + 3x_2^2 + 3x_{N-1}^2 + 4x_N^2 + 4\sum_{i=3}^{N-2} x_i^2 - 2\sum_{i=2}^{N-1}\left(x_{i-1}^2 + x_{i+1}^2\right) + 2\sum_{i=2}^{N-1}\left(x_{i-1}^2 + x_{i+1}^2\right)$$
$$- \sum_{2}^{N-1}\left(x_{i-1}^2 + x_{i+1}^2 + 2x_{i-1}x_{i+1}\right) \geq 0$$

$$\Rightarrow 32\sum_{i=1}^{N} x_i^2 + 4x_1^2 + 3x_2^2 + 3x_{N-1}^2 + 4x_N^2 + 4\sum_{i=3}^{N-2} x_i^2 - 2\sum_{i=2}^{N-1} x_{i-1}^2 - 2\sum_{i=2}^{N-1} x_{i+1}^2 + \sum_{i=2}^{N-1}(x_{i-1}+x_{i+1})^2 \geq 0$$

$$\Rightarrow 32\sum_{i=1}^{N} x_i^2 + 4x_1^2 + 3x_2^2 + 3x_{N-1}^2 + 4x_N^2 + 4\sum_{3}^{N-2} x_i^2 - 2\sum_{1}^{N-2} x_i^2 - 2\sum_{3}^{N} x_i^2 + \underbrace{\sum_{i=2}^{N-1}(x_{i-1}+x_{i+1})^2}_{(\cdot)} \geq 0$$

$$\Rightarrow 32\sum_{i=1}^{N} x_i^2 + 4x_1^2 + 3x_2^2 + 3x_{N-1}^2 + 4x_N^2 + 2\sum_{3}^{N-2} x_i^2 + 2\sum_{3}^{N-2} x_i^2 - 2\sum_{1}^{N-2} x_i^2 - 2\sum_{3}^{N} x_i^2 + (\cdot) \geq 0$$

$$\Rightarrow 32\sum_{i=1}^{N} x_i^2 + 2x_1^2 + x_2^2 + x_{N-1}^2 + 2x_N^2 + \cancel{2\sum_{1}^{N-2} x_i^2} + \cancel{2\sum_{3}^{N} x_i^2} - \cancel{2\sum_{1}^{N-2} x_i^2} - \cancel{2\sum_{3}^{N} x_i^2} + (\cdot) \geq 0$$

$$\Rightarrow 32\sum_{i=1}^{N} x_i^2 + 2x_1^2 + x_2^2 + x_{N-1}^2 + 2x_N^2 + \sum_{i=2}^{N-1}(x_{i-1}+x_{i+1})^2 \geq 0 \checkmark$$

The above relationship only equals 0 when $x = \bar{0}$

Thus, $\|Ax\|$ only equals 0 when $x = \bar{0}$, signifying that A is in fact <u>invertible</u>.

5e) $f(t) = \sum_{n=-\infty}^{\infty} a_n b_2(t-n)$

Show that there is a convolution operator that maps the sequence $\{a_n\}$ to the sequence $\{f(n)\}$. That is, find a sequence of numbers $\{h_n\}_{n\in\mathbb{Z}}$ such that

$$f(n) = \sum_{\ell=-\infty}^{\infty} h_\ell a_{n-\ell}$$

Let us first equate the two equations when $t=n$, in other words, when our domain for $f(t)$ and subsequently $b_2(t)$ is $\mathbb{Z}$ (integers).

$f(t) = \sum_{g=-\infty}^{\infty} a_g b_2(t-g)$ change indexing variable

$f(n) = \sum_{\ell=-\infty}^{\infty} a_{n-\ell} h_\ell$

From 5c) we know that for an integer domain $b_2(\cdot)$ takes on 1 of three values: $0, \frac{1}{8}, \frac{3}{4}$. $\sum_{g=-\infty}^{\infty} a_g b_2(t-g)$

$f(0) = \dots + a_{-2} b_2(0+2) + a_1 b_2(0+1) + a_0 b_2(0+0) + a_1 b_2(0-1) + a_2 b_2(0-2) + \dots$

$0 + a_{-2}(0) + a_{-1}\frac{1}{8} + a_0\frac{3}{4} + a_1\frac{1}{8} + a_2 0 + 0$

$\sum_{\ell=-\infty}^{\infty} a_{n-\ell} h_\ell$

$f(0) = \dots + a_{0+2} h_2 + a_{0+1} h_1 + a_0 h_0 + a_{0-1} h_1 + a_{0-2} h_2 + \dots$

$\dots + a_2 h_{-2} + a_1 h_{-1} + a_0 h_0 + a_{-1} h_1 + a_{-2} h_2 + \dots$

Equating the two expressions we see that there is a correspondence between $b_2(t-g)$ and $h_\ell$. In particular $a_g b_2(t-g) = a_{n-\ell} h_\ell$ when $t=n$

$a_1 h_{-1} = a_1 b_2(0-1) = a_1 \frac{1}{8}$

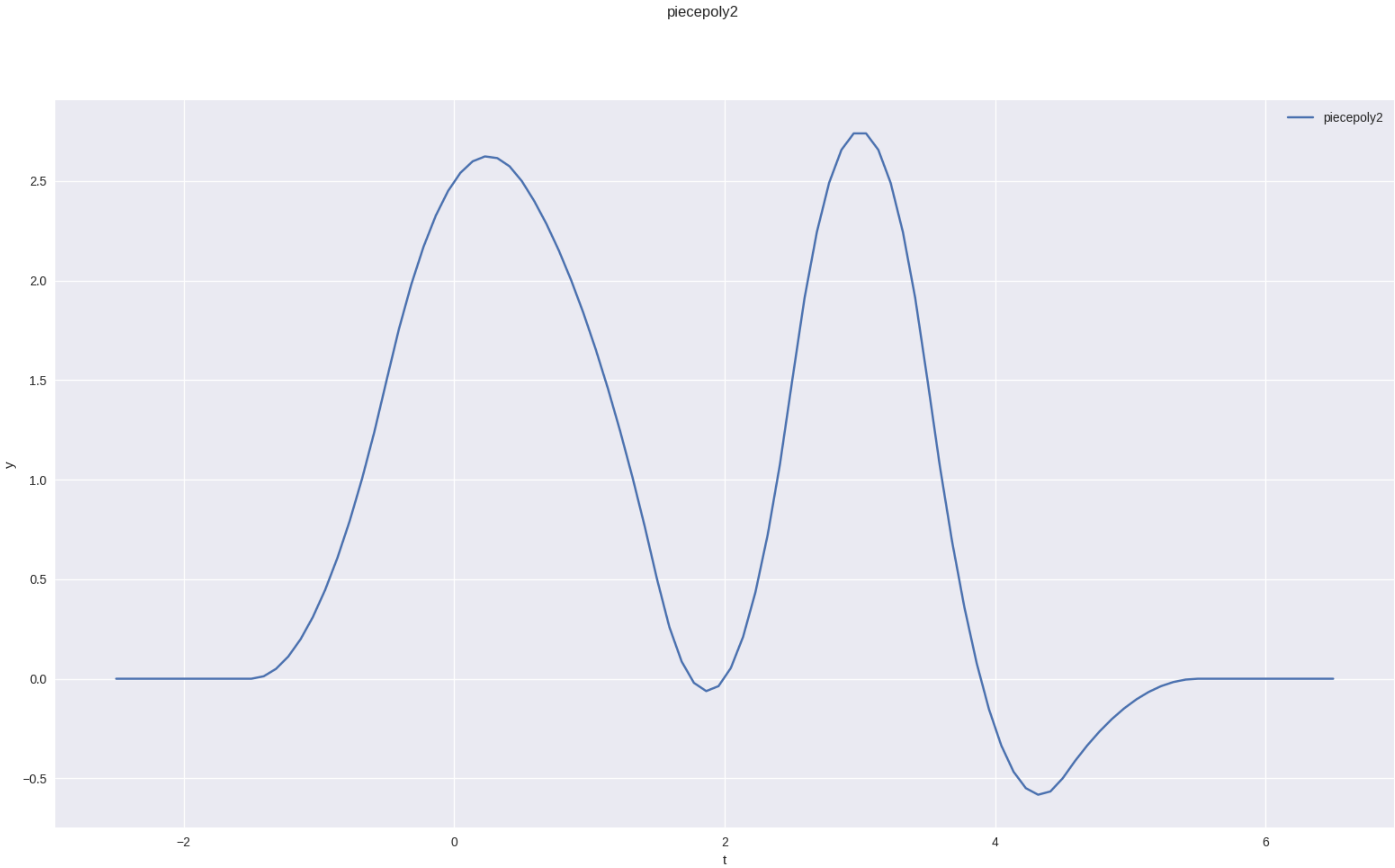
$a_0 h_0 = a_0 b_2(0+0) = a_0 \frac{3}{4}$

$a_{-1} h_1 = a_{-1} b_2(0+1) = a_{-1}\frac{1}{8}$

All other terms for $b_2$ evaluate to 0.

From this analysis, we see that the sequence $\{h_\ell\}$ is defined as follows.

$$\boxed{h_\ell = \begin{cases} \frac{1}{8} & \ell = -1, 1 \\ \frac{3}{4} & \ell = 0 \\ 0 & \text{elsewhere} \end{cases}}$$

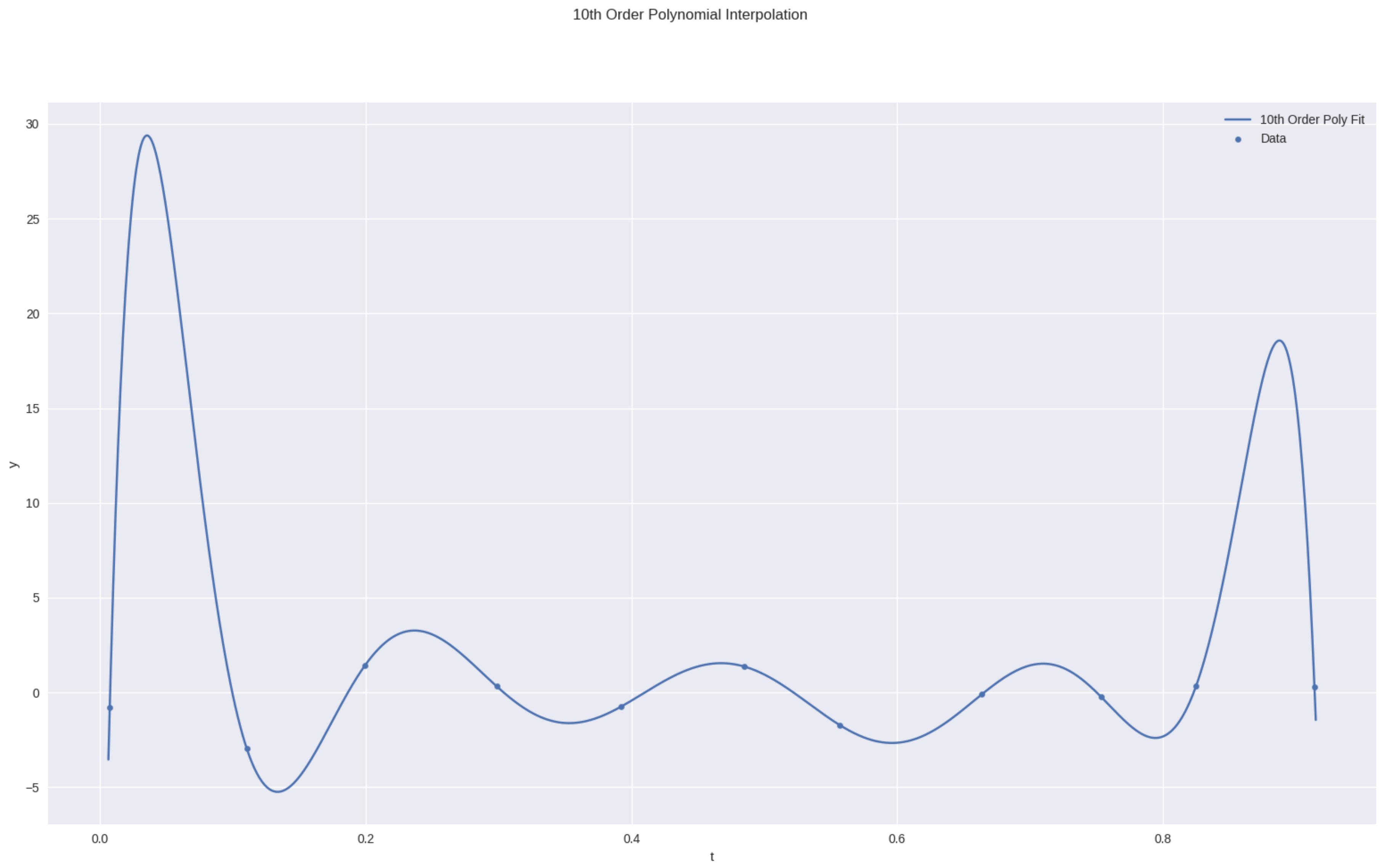

piecepoly2
piecepoly2
y
t
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
−2
0
2
4
6

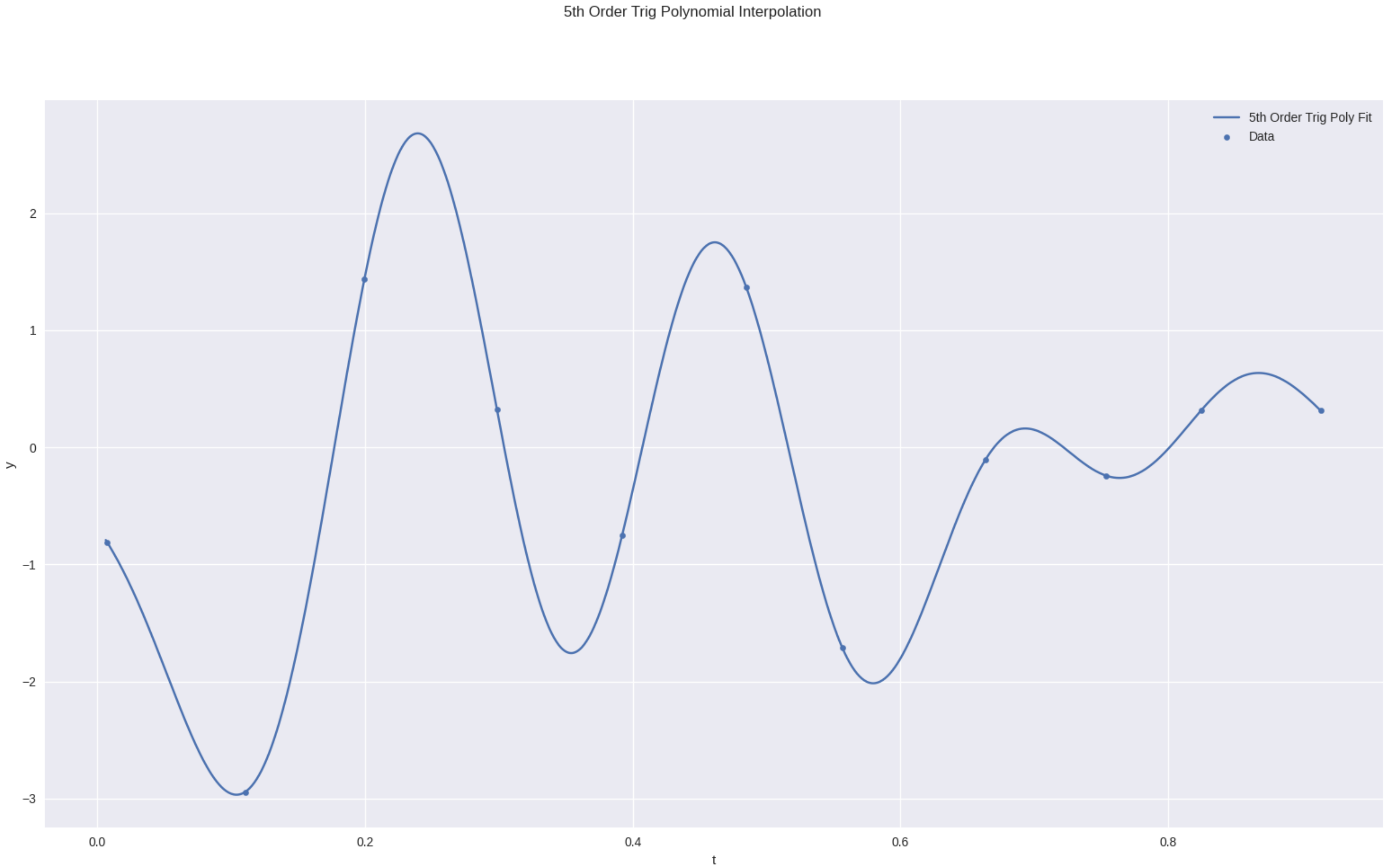
```
import sys
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
import scipy.io as sio

mpl.style.use('seaborn')

mat_filename = "hw01p6_nonuniform_samples.mat"
nonuniform_samples = sio.loadmat(mat_filename)

t_vec = nonuniform_samples['t']
y_vec = nonuniform_samples['y']

def part_a(t_vec, y_vec):
    print("""
    Find 10th order polynomial that interpolates data
    Formulate the problem as a system of equations, Ax = b
    => x = inv(A) * b
    Where b = y_vec^T and x = 10th polynomial coefficients
    """)

    A = np.zeros(shape=(len(t_vec), len(t_vec)))
    b = np.zeros(shape=(len(t_vec), 1))

    for i in range(0, len(t_vec)):
        t = t_vec[i]
        y = y_vec[i]

        A[i,:] = [t**10, t**9, t**8, t**7, t**6, t**5, t**4, t**3, t**2, t**1,
1]
        b[i,:] = [y]

    print("A: ")
    print(A)

    print("b: ")
    print(b)

    x = np.linalg.inv(A) @ b
    print("x (coefficients): ")
    print(x)

    # Plot result with data points overlaid
    fig = plt.figure()
    fig.suptitle("10th Order Polynomial Interpolation")
    ax = fig.add_subplot(111)
    ax.scatter(t_vec, y_vec, s = 20, label="Data")
    t_domain = np.linspace(min(t_vec) - 0.001, max(t_vec) + 0.001, 1000)
    ax.plot(t_domain, [np.polyval(x, i) for i in t_domain], label="10th Order
Poly Fit")

    ax.set_xlabel("t")
    ax.set_ylabel("y")
    ax.legend()

    plt.show()

part_a(t_vec, y_vec)

# Helper function to evaluate a generic trigonometric polynomial
# Coefficient ordering should be [ao, a1, ..., an, b1, ..., bn]
# Where n is the order
def trigval(coeffs, t):
```

```
    val = coeffs[0].copy()
    order = len(coeffs)//2
    for k in range(1, order+1):
        val += coeffs[k] * np.cos(2 * np.pi * k * t) \
                + coeffs[order + k] * np.sin(2 * np.pi * k * t)
    return val

def part_b(t_vec, y_vec):

    print("""
    Find 5th order trig polynomial that interpolates data
    Formulate the problem as a system of equations, Ax = b
    => x = inv(A) * b
    Where b = y_vec^T and x = 5th trig polynomial coefficients
    """)

    A = np.zeros(shape=(len(t_vec), len(t_vec)))
    b = np.zeros(shape=(len(t_vec), 1))

    for i in range(0, len(t_vec)):
        t = t_vec[i]
        y = y_vec[i]

        A[i,:] = [1, np.cos(2 * np.pi * 1 * t), np.cos(2 * np.pi * 2 * t),
                np.cos(2 * np.pi * 3 * t), np.cos(2 * np.pi * 4 * t),
                np.cos(2 * np.pi * 5 * t), np.sin(2 * np.pi * 1 * t),
                np.sin(2 * np.pi * 2 * t), np.sin(2 * np.pi * 3 * t),
                np.sin(2 * np.pi * 4 * t), np.sin(2 * np.pi * 5 * t)]
        b[i,:] = [y]

    print("A: ")
    print(A)

    print("b: ")
    print(b)

    x = np.linalg.inv(A) @ b
    print("x (coefficients): ")
    print(x)

    # Plot result with data points overlaid
    fig = plt.figure()
    fig.suptitle("5th Order Trig Polynomial Interpolation")
    ax = fig.add_subplot(111)
    ax.scatter(t_vec, y_vec, s = 20, label="Data")
    t_domain = np.linspace(min(t_vec) - 0.001, max(t_vec) + 0.001, 1000)
    ax.plot(t_domain, [trigval(x, i) for i in t_domain], label="5th Order Trig
Poly Fit")

    ax.set_xlabel("t")
    ax.set_ylabel("y")
    ax.legend()

    plt.show()

part_b(t_vec, y_vec)
```

```
    Find 10th order polynomial that interpolates data
    Formulate the problem as a system of equations, Ax = b
    => x = inv(A) * b
    Where b = y_vec^T and x = 10th polynomial coefficients

A:
[[5.44573244e-22 7.28535418e-20 9.74641815e-18 1.30388536e-15
  1.74435060e-13 2.33360932e-11 3.12192539e-09 4.17654235e-07
  5.58741925e-05 7.47490418e-03 1.00000000e+00]
 [2.71399899e-10 2.45611509e-09 2.22273530e-08 2.01153122e-07
  1.82039573e-06 1.64742192e-05 1.49088406e-04 1.34922042e-03
  1.22101763e-02 1.10499667e-01 1.00000000e+00]
 [1.00579869e-07 5.03802085e-07 2.52353222e-06 1.26403106e-05
  6.33150043e-05 3.17143295e-04 1.58856294e-03 7.95707259e-03
  3.98567804e-02 1.99641630e-01 1.00000000e+00]
 [5.63124118e-06 1.88600880e-05 6.31659893e-05 2.11554803e-04
  7.08536906e-04 2.37302364e-03 7.94770340e-03 2.66183566e-02
  8.91498929e-02 2.98579793e-01 1.00000000e+00]
 [8.57002441e-05 2.18616961e-04 5.57680742e-04 1.42261519e-03
  3.62901895e-03 9.25744263e-03 2.36152650e-02 6.02413392e-02
  1.53672590e-01 3.92010957e-01 1.00000000e+00]
 [7.16184915e-04 1.47748354e-03 3.04803630e-03 6.28807363e-03
  1.29722438e-02 2.67616314e-02 5.52090237e-02 1.13895758e-01
  2.34966006e-01 4.84732922e-01 1.00000000e+00]
 [2.84850474e-03 5.11864785e-03 9.19800323e-03 1.65284399e-02
  2.97009382e-02 5.33713851e-02 9.59062210e-02 1.72339601e-01
  3.09687296e-01 5.56495549e-01 1.00000000e+00]
 [1.65480927e-02 2.49386621e-02 3.75835982e-02 5.66400414e-02
  8.53588917e-02 1.28639390e-01 1.93864897e-01 2.92162441e-01
  4.40300916e-01 6.63551743e-01 1.00000000e+00]
 [5.89790764e-02 7.82759182e-02 1.03886323e-01 1.37875969e-01
  1.82986386e-01 2.42856082e-01 3.22314013e-01 4.27769080e-01
  5.67727058e-01 7.53476647e-01 1.00000000e+00]
 [1.45508289e-01 1.76440777e-01 2.13948964e-01 2.59430728e-01
  3.14581110e-01 3.81455488e-01 4.62546175e-01 5.60875307e-01
  6.80107473e-01 8.24686288e-01 1.00000000e+00]
 [4.06212274e-01 4.44506112e-01 4.86409931e-01 5.32264044e-01
  5.82440847e-01 6.37347843e-01 6.97430951e-01 7.63178123e-01
  8.35123315e-01 9.13850816e-01 1.00000000e+00]]
b:
[[-0.80949869]
 [-2.94428416]
 [ 1.43838029]
 [ 0.32519054]
 [-0.75492832]
 [ 1.37029854]
 [-1.71151642]
 [-0.10224245]
 [-0.24144704]
 [ 0.31920674]
 [ 0.3128586 ]]
x (coefficients):
[[-7.17863977e+06]
 [ 3.36128732e+07]
 [-6.75821318e+07]
 [ 7.62617070e+07]
 [-5.29648340e+07]
 [ 2.33434760e+07]
 [-6.48638637e+06]
 [ 1.08866072e+06]
 [-9.99790912e+04]
 [ 3.97584708e+03]
 [-2.53772930e+01]]
```

```
    Find 5th order trig polynomial that interpolates data
    Formulate the problem as a system of equations, Ax = b
    => x = inv(A) * b
    Where b = y_vec^T and x = 5th trig polynomial coefficients

A:
[[ 1.          0.99889729  0.99559159  0.9900902   0.98240524  0.97255367
   0.04694894  0.09379434  0.14043289  0.18676172  0.23267867]
 [ 1.          0.76850826  0.18120988 -0.48998568 -0.93432596 -0.94608875
   0.63983987  0.98344445  0.87173048  0.3564197  -0.32390751]
 [ 1.          0.31115771 -0.80636176 -0.81296906  0.30043858  0.99993662
   0.95035829  0.59142262 -0.58230688 -0.95380116 -0.01125829]
 [ 1.         -0.30051812 -0.81937771  0.79299383  0.34275968 -0.99900482
   0.9537761  -0.57325401 -0.60922966  0.93942312  0.04460231]
 [ 1.         -0.77850551  0.21214167  0.4481986  -0.90999183  0.96866871
   0.62763777 -0.97723892  0.89393401 -0.41462619 -0.24835647]
 [ 1.         -0.99540264  0.98165283 -0.958877    0.92728456 -0.88716599
   0.09577883 -0.19067701  0.28382196 -0.37435725  0.46145043]
 [ 1.         -0.93765622  0.75839838 -0.48457769  0.1503362   0.20265035
  -0.34756411  0.6517913  -0.87474823  0.98863493 -0.97925116]
 [ 1.         -0.5168527  -0.46572658  0.99827677 -0.5661975  -0.41299536
  -0.85607435  0.88492867 -0.05868119 -0.82426961  0.91073313]
 [ 1.          0.02184268 -0.99904579 -0.06548636  0.996185    0.10900507
  -0.99976142 -0.04367494  0.99785346  0.08726654 -0.99404119]
 [ 1.          0.45223334 -0.59097    -0.98674603 -0.30150891  0.71404126
  -0.89189966 -0.80669353  0.16227223  0.95346336  0.70010362]
 [ 1.          0.85704407  0.46904908 -0.05305261 -0.55998593 -0.90681263
  -0.51524311 -0.8831721  -0.99859172 -0.82850212 -0.42153394]]
b:
[[-0.80949869]
 [-2.94428416]
 [ 1.43838029]
 [ 0.32519054]
 [-0.75492832]
 [ 1.37029854]
 [-1.71151642]
 [-0.10224245]
 [-0.24144704]
 [ 0.31920674]
 [ 0.3128586 ]]
x (coefficients):
[[-0.27318033]
 [-0.280127  ]
 [-0.56582526]
 [-0.4430021 ]
 [ 0.8882698 ]
 [-0.00691239]
 [ 0.06352766]
 [-0.76114309]
 [-0.5603032 ]
 [-0.83318278]
 [ 0.78315909]]
```


10th Order Polynomial Interpolation
10th Order Poly Fit
Data
y
t
30
25
20
15
10
5
0
−5
0.0
0.2
0.4
0.6
0.8


5th Order Trig Polynomial Interpolation
5th Order Trig Poly Fit
Data
2
1
0
−1
−2
−3
y
0.0
0.2
0.4
0.6
0.8
t