For the source code for the coding problems, see the attached Jupyter notebook, Matlab live script. Alternatively, the entire git repository is attached as a zip archive, and is available on GitHub. The comments in the code have been omitted here for brevity. They are present in the Jupyter notebook.

(1) **Problem Statement:** Compute the interpolating polynomial $p_2(x)$ that interpolates $f = \sqrt{2}xcos(x)$ at points $x_0 = 0$, $x_1 = \frac{\pi}{4}$, and $x_2 = \frac{\pi}{2}$ in the interval $[0, \frac{\pi}{2}]$.

$$\begin{aligned} a(x_0)^2 + b(x_0) + c &= 0 \\ a(x_1)^2 + b(x_1) + c &= \frac{\pi}{4} \\ a(x_2)^2 + b(x_2) + c &= 0 \end{aligned}$$

We can solve this as a matrix-vector equation.

$$\begin{bmatrix} 0 & 0 & 1 \\ \frac{\pi^2}{16} & \frac{\pi}{4} & 1 \\ \frac{\pi^2}{4} & \frac{\pi}{2} & 1 \end{bmatrix} \times \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{\pi}{4} \\ 0 \end{bmatrix}$$

$$\left[\begin{array}{ccc|c} 0 & 0 & 1 & 0 \\ \frac{\pi^2}{16} & \frac{\pi}{4} & 1 & \frac{\pi}{4} \\ \frac{\pi^2}{4} & \frac{\pi}{2} & 1 & 0 \end{array}\right] \text{swap(R1, R3)}$$

$$\left[\begin{array}{ccc|c} \frac{\pi^2}{4} & \frac{\pi}{2} & 1 & 0 \\ \frac{\pi^2}{16} & \frac{\pi}{4} & 1 & \frac{\pi}{4} \\ 0 & 0 & 1 & 0 \end{array}\right] \text{swap(R1, R2)}$$

$$\left[\begin{array}{ccc|c} \frac{\pi^2}{16} & \frac{\pi}{4} & 1 & \frac{\pi}{4} \\ \frac{\pi^2}{4} & \frac{\pi}{2} & 1 & 0 \\ 0 & 0 & 1 & 0 \end{array}\right] \text{R2} = \text{R2} - 4 \times \text{R1}$$

$$\left[\begin{array}{ccc|c} \frac{\pi^2}{16} & \frac{\pi}{4} & 1 & \frac{\pi}{4} \\ 0 & -\frac{\pi}{2} & -3 & -\pi \\ 0 & 0 & 1 & 0 \end{array}\right]$$

Now we can solve for a, b, and c.

$$\begin{aligned} 1(c) &= 0 \\ c &= 0 \end{aligned}$$

$$\begin{aligned} -\frac{\pi}{2}b - 3c &= -\pi \\ -\frac{\pi}{2}b &= -\pi \\ b &= 2 \end{aligned}$$

$$\frac{\pi^2}{16}a + \frac{\pi}{4}b + 1c = \frac{\pi}{4}$$
$$\frac{\pi^2}{16}a + \frac{\pi}{4}2 + 1(0) = \frac{\pi}{4}$$
$$\frac{\pi^2}{16}a + \frac{\pi}{2} = \frac{\pi}{4}$$
$$\frac{\pi^2}{16}a = -\frac{\pi}{4}$$
$$\pi^2 a = -4\pi$$
$$a = \frac{-4}{\pi}$$

Now that we have the coefficients, we can put them together to find the Lagrange polynomial $p_2(x) = -\frac{4}{\pi}x^2 + 2x$

(2) **Problem Statement:** U.S. populations for the years 1990, 2000, and 2010, is given in the table

x (year)	1990	2000	2010
y (population)	$248,709,873$	$281,421,906$	$308,745,538$

Estimate the population in 2006 using

(a) Linear Splines

$$\begin{aligned} y(2006) &\approx \frac{y(2010) - y(2000)}{2010 - 2000}(2006 - 2000) + 281,421,906 \\ &\approx \frac{308,745,538 - 281,421,906}{10}6 + 281,421,906 \\ &\approx \frac{27,323,632}{10}6 + 281,421,906 \\ &\approx 2,732,363.2 \times 6 + 281,421,906 \\ y(2006) &\approx 297,816,085 \end{aligned}$$

(b) $p_2(x)$ that interpolates the population of the U.S. at the data points $x_0 = 1990$, $x_1 = 2000$, and $x_2 = 2010$.

$$\left[\begin{array}{ccc|c} 3,960,100 & 1990 & 1 & 248,709,873 \\ 4,000,000 & 2000 & 1 & 281,421,906 \\ 4,040,100 & 2010 & 1 & 308,745,538 \end{array}\right] \begin{array}{l} \text{R2} = \text{R2} - 1.010075503\text{R1} \\ \text{R3} = \text{R3} - 1.020201510\text{R1} \end{array}$$

$$\left[\begin{array}{ccc|c} 3,960,100 & 1990 & 1 & 248,709,873 \\ 0 & -10.05025097 & -0.010075503 & 30,206,155.93 \\ 0 & -20.2010049 & -0.020201510 & 55,011,350.01 \end{array}\right] \text{R3} = \text{R3} - 2.010000045\text{R2}$$

$$\left[\begin{array}{ccc|c} 3,960,100 & 1990 & 1 & 248,709,873 \\ 0 & -10.05025097 & -0.010075503 & 30,206,155.93 \\ 0 & 0 & 0.00005025148340 & -5,703,024.769 \end{array}\right]$$

$$0.00005025148340c = -5703024.769$$
$$c = -113489679968.33304$$

$$\begin{aligned}
-10.050250097b - 0.010075503c &= 30206155.93\\
-10.050250097b + 1143465610.9899793 &= 30206155.93\\
-10.050250097b &= -1113259455.0599792\\
b &= 110769318.93373197
\end{aligned}$$

$$\begin{aligned}
3960100a + 1990b + c &= 248709873\\
3960100a + 220430944678.12662 - 113489679968.33304 &= 2487098733960100a \qquad = -106692554836.79358\\
a &= -26941.884002119536
\end{aligned}$$

$$\begin{aligned}
&p_2(x) = -26941.884x^2 + 110769318.93373x - 113489679968.33304\\
&p_2(2006) = 298,462,693
\end{aligned}$$

(3) **Problem Statement:** Find b, c, and d such that

$$s(x) = \begin{cases} 1 + 2x - x^3, & 0 \le x \le 1 \\ 2 + b(x-1) + c(x-1)^2 + d(x-1)^3, & 1 < x \le 2 \end{cases}$$

is a natural cubic spline.

$$s'(x) = \begin{cases} 2 - 3x^2, & 0 \le x \le 1 \\ b + 2c(x-1) + 3d(x-1)^2, & 1 < x \le 2 \end{cases}$$

$$\begin{aligned}
\lim_{x\to 1^+} s'(x) = \lim_{x \to 1^-} s'(x) \implies 2 - 3(1)^2 &= b - 2c(1-1) + 3d(1-1)^2\\
2 - 3 &= b - 2c(0) + 3d(0)^2\\
-1 &= b\\
b &= -1
\end{aligned}$$

$$s''(x) = \begin{cases} -6x, & 0 \le x \le 1 \\ 2c + 6d(x-1), & 1 < x \le 2 \end{cases}$$

$$\begin{aligned}
\lim_{x\to 1^+} s''(x) = \lim_{x \to 1^-} s''(x) \implies -6(1) &= 2c + 6d(1-1)\\
-6 &= 2c\\
c &= -3
\end{aligned}$$

$$\begin{aligned}
s''(0) = s''(2) = 0 \implies -6(0) = 2(-3) + 6d(2-1) &= 0\\
-6 + 6d(1) &= 0\\
6d &= 6\\
d &= 1
\end{aligned}$$

(4) **Problem Statement:** The following function s is a cubic spline that interpolates the following values. Find a, b, c, d, y_1, and y_2.

$$s(x) = \begin{cases} \frac{5}{4}(x+2)^3 - \frac{9}{2}(x+2)^2 + 8, & -2 \le x < 0 \\ ax^3 + bx^2 + cx + d, & 0 \le x < 1 \\ (x-1)^3 - 1, & 1 \le x < 2 \\ -\frac{5}{4}(x-2)^3 + 3(x-2)^2 + 3(x-2), & 2 \le x \le 4 \end{cases}$$

$$s'(x) = \begin{cases} \frac{15}{4}(x+2)^2 - 9(x+2), & -2 \le x < 0 \\ 3ax^2 + 2bx + c, & 0 \le x < 1 \\ 3(x-1)^2, & 1 \le x < 2 \\ -\frac{15}{4}(x-2)^2 + 6(x-2), & 2 \le x \le 4 \end{cases}$$

$$s''(x) = \begin{cases} \frac{15}{2}(x+2) - 9, & -2 \le x < 0 \\ 6ax + 2b, & 0 \le x < 1 \\ 6(x-1), & 1 \le x < 2 \\ -\frac{15}{2}(x-2) + 6, & 2 \le x \le 4 \end{cases}$$

x	-2	0	1	2	4
y	8	y_1	y_2	0	8

$$\lim_{x\to 0^+} s''(x) = \lim_{x\to 0^-} s''(x) \implies \frac{15}{2}(0+2) - 9 = 6a(0) + 2b$$
$$15 - 9 = 2b$$
$$6 = 2b$$
$$b = 3$$

$$\lim_{x\to 1^+} s''(x) = \lim_{x\to 1^-} s''(x) \implies 6(1-1) = 6a(1) + 2b$$
$$0 = 6a + 2(3)$$
$$-6 = 6a$$
$$a = -1$$

$$\lim_{x\to 1^+} s'(x) = \lim_{x\to 1^-} s'(x) \implies 3a(1)^2 + 2b(1) + c = 3(1-1)^2$$
$$3(-1) + 2(3) + c = 3(0)$$
$$6 - 3 + c = 0$$
$$3 + c = 0$$
$$c = -3$$

$$\lim_{x\to 1^+} s(x) = \lim_{x\to 1^-} s(x) \implies a(1)^3 + b(1)^2 + c(1) + d = (1-1)^3 - 1$$
$$-1 + 3 - 3 + d = -1$$
$$d = 0$$

$$\begin{aligned} y_1 = s(0) &= ax^3 + bx^2 + cx + d \\ &= -1(0)^3 + 3(0)^2 - 3(0) + 0 \\ y_1 &= 0 \end{aligned}$$

$$\begin{aligned} y_2 = s(1) &= (x-1)^3 - 1 \\ &= (1-1)^3 - 1 \\ y_2 &= -1 \end{aligned}$$

(5) **Problem Statement:** Fine the line of best fit for the following data. What is the error? What does it mean?

i	x_i	y_i
1	2	3
2	4	7
3	6	11

$$\begin{bmatrix} 2 & 1 \\ 4 & 1 \\ 6 & 1 \end{bmatrix} \times \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 3 \\ 7 \\ 11 \end{bmatrix}$$

$$\begin{bmatrix} 2 & 4 & 6 \\ 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 2 & 1 \\ 4 & 1 \\ 6 & 1 \end{bmatrix} \times \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 2 & 4 & 6 \\ 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 3 \\ 7 \\ 11 \end{bmatrix}$$

$$\begin{bmatrix} 56 & 12 \\ 12 & 3 \end{bmatrix} \times \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 100 \\ 21 \end{bmatrix}$$

$$\left[\begin{array}{cc|c} 56 & 12 & 100 \\ 12 & 3 & 21 \end{array}\right] \text{swap(R1, R2)}$$

$$\left[\begin{array}{cc|c} 12 & 3 & 21 \\ 56 & 12 & 100 \end{array}\right] \text{R2} = \text{R2} - \frac{14}{3}\text{R1}$$

$$\left[\begin{array}{cc|c} 12 & 3 & 21 \\ 0 & -2 & 2 \end{array}\right]$$

$$\begin{aligned} -2a_2 &= 2 \\ a_2 &= -1 \end{aligned}$$

$$\begin{aligned} 12a_1 + 3a_2 &= 21 \\ 12a_1 - 3 &= 21 \\ 12a_1 &= 24 \\ a_1 &= 2 \end{aligned}$$

The line of best fit is $y = 2x - 1$. The error is $\sum_{i=1}^{3}(y_i - P(x_i))^2 = 0$. This means that the line exactly matches the data points. The error cannot "cancel out" since it is squared, and all of the terms have the same sign.

(6) **Problem Statement:** Write a `Matlab` function, called `Lagrange_poly` that inputs a set of data points $(x, y) =$ (`datx, daty`), a set x of numbers at which to interpolate, and outputs the polynomial interpolant, y, evaluated at x using Lagrange polynomial interpolation.

(a) Use the code to interpolate the following functions:

(i) $f_1(x) = e^{-x^2}$

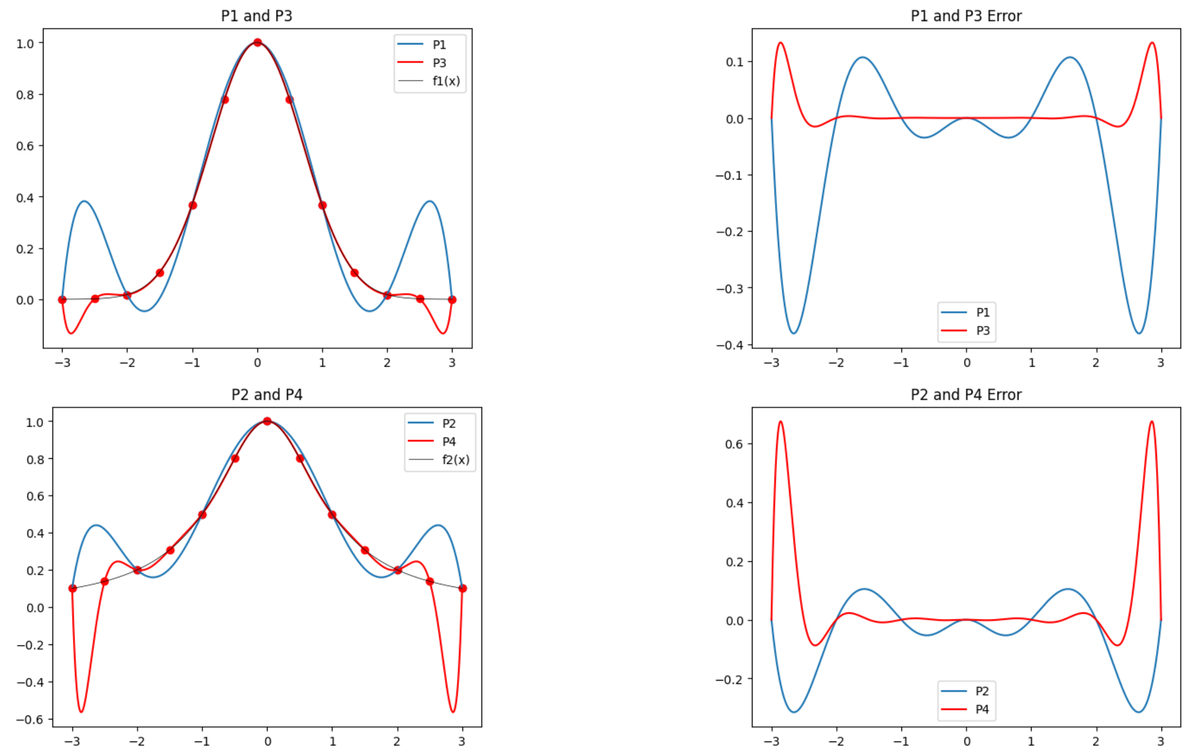
(ii) $f_2(x) = \frac{1}{1+x^2}$

using the data points `datx=-3:1:3`. Interpolate at the points `x=-3:0.01:3`. Call P_1 the Lagrange interpolant of f_1 and P_2 the Lagrange interpolant of f_2. Repeat the experiment except using the data `datx1=-3:0.5:3`. Call in that case P_3 and P_4 the new interpolants. Compare the results.

For each interpolation problem, plot on the same graph the function, the two interpolants, and the data set. On a separate plot, plot the error between each interpolant `y` and `f(x)`.

```
def Lagrange_interp(x_data, y_data, x_interp):
    n = len(x_data)
    m = len(y_data)
    xs = len(x_interp)
    if(n != m):
        return
    p_interp = x_interp * 0
    for i in range(n):
        L_i = np.array([1.] * xs)
        for j in range(n):
            if(i == j):
                continue
            L_i *= (x_interp - x_data[j]) / (x_data[i] - x_data[j])
        p_interp += y_data[i] * L_i
    return(p_interp)
```

LISTING 1. Python



(7) **Problem Statement:** Write a `Matlab` function called `linear_spline` which inputs a set of data points $(x, y) =$ `(datx, daty)`, a set x of numbers at which to interpolate, and outputs the polynomial interpolant, y, evaluated at x using a linear spline of interpolation.

(a) Use the code to interpolate the following functions

(i) $f_1(x) = e^{-x^2}$

(ii) $f_2(x) = \frac{1}{1+x^2}$

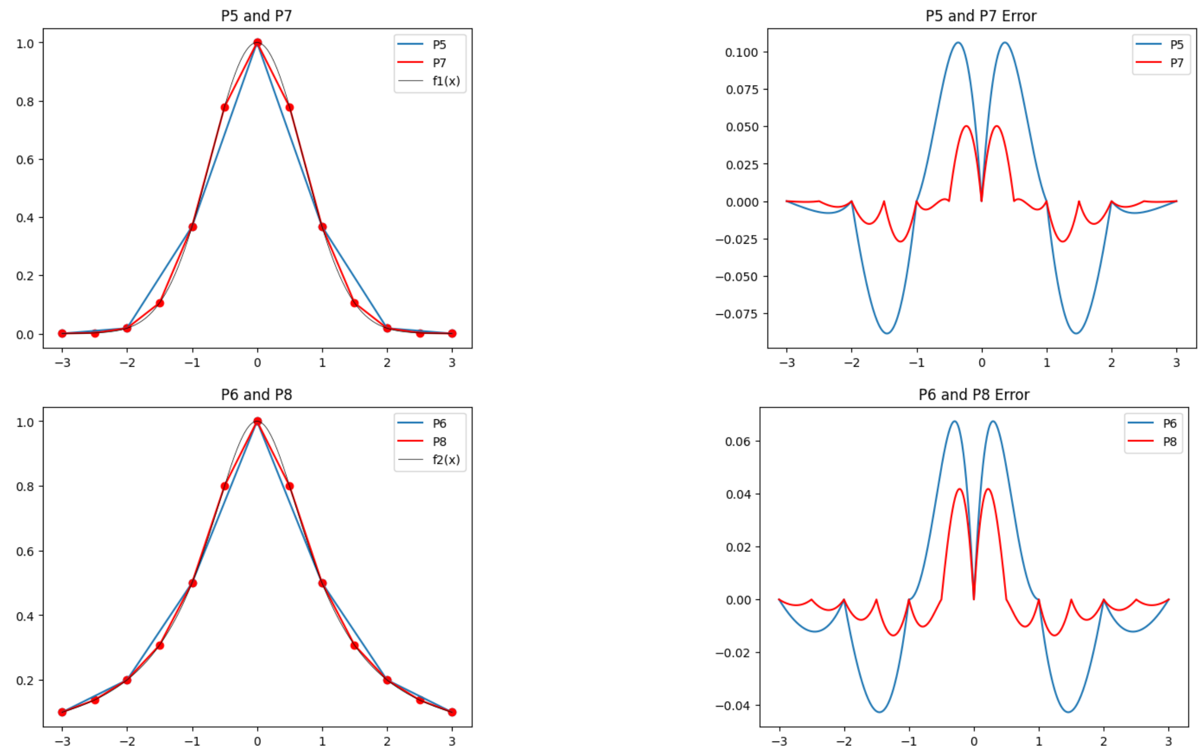
using the data points `datx=-3:1:3`. Interpolate at the points `x=-3:0.01:3`. Call P_5 the Linear Spline interpolant of f_1 and P_6 the Linear Spline interpolant of f_2. Repeat the experiment except using the data `datx1=-3:0.5:3`. Call in that case P_7 and P_8 the new interpolants. Compare the results.

For each interpolation problem, plot on the same graph the function, the two interpolants, and the data set. On a separate plot, plot the error between each interpolant `y` and `f(x)`.

```
def linear_spline(x_data, y_data, x_interp):
    if(len(x_data) != len(y_data)):
        return
    p_interp = x_interp.copy()
    for i, x in enumerate(x_interp):
        dist = [(xd, yd) for xd, yd in zip(x_data, y_data)]
        dist.sort(key=lambda k: abs(k[0] - x))
        important = dist[:2]
        lowerx, lowery = min(important)
        higherx, highery = max(important)
        p_interp[i] = (highery - lowery) / (higherx - lowerx) * (x -
  lowerx) + lowery

    return(p_interp)
```

Listing 2. Python



(8) **Problem Statement:** Write a `Matlab` function called `least_squares` which inputs a set of data points $(x, y) =$ `(datx, daty)`, the degree of polynomial n, and outputs the coefficients a_i as a vector.
Write a second function called `exp_least_squares` which computes the exponential components for least squares as $p_2(x) = a_0 e^{a_1 x}$ and outputs $a = (a_0, a_1)$
Write also at the top of the script an anonymous function `N(d)` which uses a normal distribution.
To create `daty` in each case, add `N(0.5)` to each data point.
(a) Use the code to approximate the functions
   (i) $f_1(x) = 2x + 4$ on $[0, 4]$ using $n = 1$.
   (ii) $f_2(x) = x^2 - 3x + 1$ on $[1, 5]$ using $n = 2$.
(b) Use the code to approximate
   (i) $f_3(x) = 3e^{0.25x}$ on $[0, 10]$

```
def least_squares(datx, daty, n):
    X = np.vander(datx, n+1)
    coefficients = np.linalg.solve(X.T @ X, X.T @ daty)
    return(coefficients)

def exp_least_squares(datx, daty):
    log_daty = np.log(daty)
    X = np.vstack([np.ones_like(datx), datx]).T
    b, _, _, _ = np.linalg.lstsq(X, log_daty, rcond=None)
    b0 = b[0]
    b1 = b[1]
    a0 = np.exp(b0)
    a1 = b1
    return(a0, a1)
```

LISTING 3. Python