1. a)

$$B = \begin{bmatrix} 1 & -1 & -1^2 \\ 1 & 0 & 0^2 \\ 1 & 1 & 1^2 \end{bmatrix}, \quad a = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix}, \quad f = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

$$\begin{aligned} a_0 - a_1 + a_2 &= 1 \\ a_0 &= 0 \\ a_0 + a_1 + a_2 &= 1 \end{aligned}$$

$$\Rightarrow a = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \Rightarrow P_n(x) = x^2$$

b) $P_2(x) = f_0 l_0(x) + f_1 l_1(x) + f_2 l_2(x)$

$$= f_0 \cdot \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} + f_1 \cdot \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} +$$

$$f_2 \cdot \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} = 1 \cdot \frac{(x-0)(x-1)}{(-1-0)(-1-1)} +$$

$$0 \cdot \frac{(x+1)(x-1)}{(0+1)(0-1)} + 1 \cdot \frac{(x+1)(x-0)}{(1+1)(1-0)} = \frac{x^2-x}{2} + \frac{x^2+x}{2}$$

$$= x^2$$

c) $b_2(x) = (x-x_0)(x-x_1)$

$b_1(x) = x - x_0$

$b_0(x) = 1$

$P_n(x) = \sum_{j=0}^{2} a_j b_j(x)$

$a_0 = f_0 = 1$

$a_1 = \dfrac{f_1 - f_0}{x_1 - x_0} = \dfrac{0-1}{0+1} = -1$

$a_2 = \dfrac{\dfrac{f_2 - f_1}{x_2 - x_1} - \dfrac{f_1 - f_0}{x_1 - x_0}}{x_2 - x_0}$

$= \dfrac{\dfrac{1-0}{1-0} - \dfrac{0-1}{0+1}}{1+1} = 1$

$P_n(x) = 1 - (x+1) + (x+1)x = x^2$

2. 5 points from $[0, \frac{\pi}{2}]$:

$$
\begin{array}{lcl}
(0, y_0) & & (0, 0) \\
(\frac{\pi}{8}, y_1) & & (\frac{\pi}{8}, \frac{\sqrt{2-\sqrt{2}}}{2}) \\
(\frac{\pi}{4}, y_2) & \Rightarrow & (\frac{\pi}{4}, \frac{\sqrt{2}}{2}) \\
(\frac{3\pi}{8}, y_3) & & (\frac{3\pi}{8}, \sqrt{\frac{1+\frac{\sqrt{2}}{2}}{2}}) \\
(\frac{\pi}{2}, y_4) & & (0, 1)
\end{array}
$$

with monomial bias:

$$
B = \begin{bmatrix} 1 & x_0 & x_0^2 & x_0^3 & x_0^4 \\ 1 & x_1 & x_1^2 & x_1^3 & x_1^4 \\ 1 & x_2 & x_2^2 & x_2^3 & x_2^4 \\ 1 & x_3 & x_3^2 & x_3^3 & x_3^4 \\ 1 & x_4 & x_4^2 & x_4^3 & x_4^4 \end{bmatrix}, \quad a = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix}, \quad f = \begin{bmatrix} 0 \\ \frac{\sqrt{2-\sqrt{2}}}{2} \\ \frac{\sqrt{2}}{2} \\ \sqrt{\frac{1+\frac{\sqrt{2}}{2}}{2}} \\ 1 \end{bmatrix}
$$

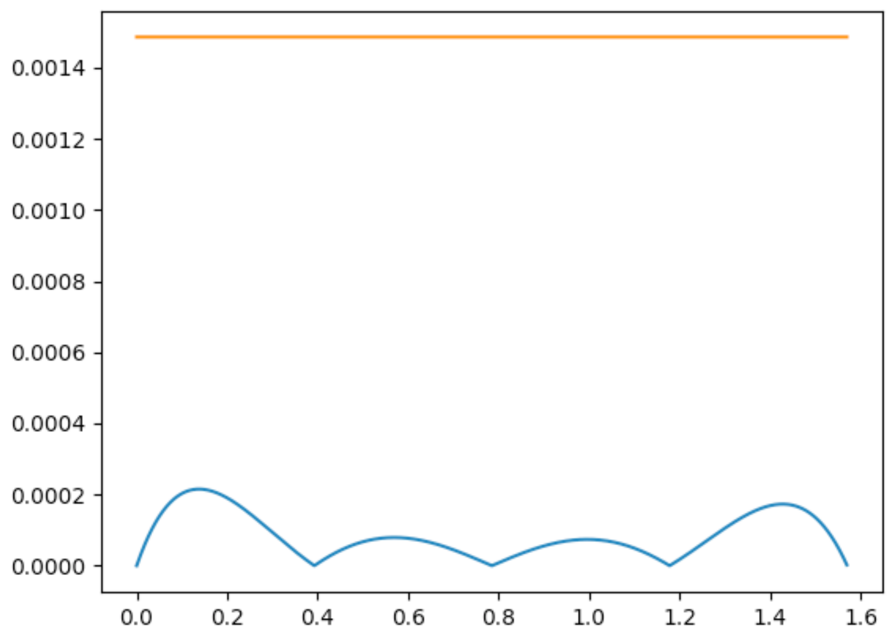
$$
= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & \frac{\pi}{8} & \frac{\pi^2}{64} & \frac{\pi^3}{512} & \frac{\pi^4}{4096} \\ 1 & \frac{\pi}{4} & \frac{\pi^2}{16} & \frac{\pi^3}{64} & \frac{\pi^4}{256} \\ 1 & \frac{3\pi}{8} & \frac{9\pi^2}{64} & \frac{27\pi^3}{512} & \frac{81\pi^4}{4096} \\ 1 & \frac{\pi}{2} & \frac{\pi^2}{4} & \frac{\pi^3}{8} & \frac{\pi^4}{16} \end{bmatrix} \Longrightarrow a = \begin{bmatrix} 0 \\ 0.99631689 \\ 0.01995143 \\ -0.20358546 \\ 0.02871423 \end{bmatrix}
$$

$$P_4(t) = 0.99631689x + 0.01995143x^2 - 0.20358546x^3 + 0.02871423x^4$$

$$\text{Bound} = \frac{Mh^n}{4n} = \frac{1 \cdot \left(\frac{\pi}{8}\right)^4}{4 \times 4} = 0.001486348$$

(the bound for $\sin(x)$'s 4th derivative is 1)

Here I plot the error as the yellow line at the top, and the absolute difference $P_4(t)$ and $\sin(x)$, which is the blue curve at the bottom. we can see that the error does not exceed the bound in the interval $[0, \frac{\pi}{2}]$



Set we want $n+1$ points, then we will use degree $n$ polynomial, thus $h$ should be $\frac{\frac{\pi}{2}}{n}$.

$$\frac{Mh^n}{4n} = 10^{-10}$$

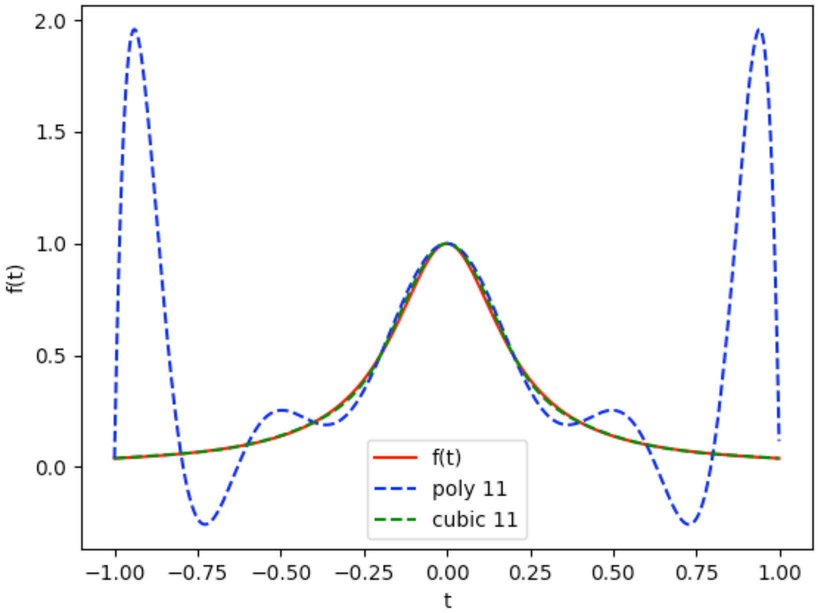
$$\frac{1 \cdot \left(\frac{\frac{\pi}{2}}{n}\right)^n}{4n} = 10^{-10}$$

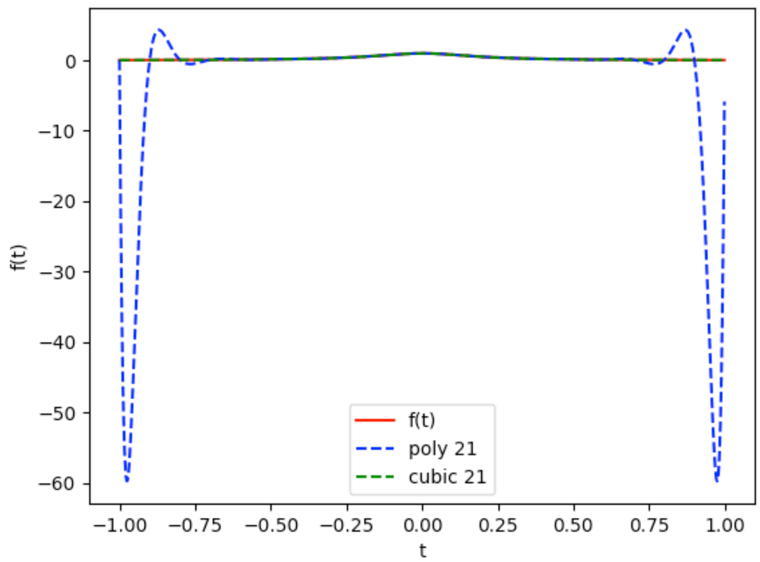
$$\frac{\pi^n}{n(2n)^n} = 4 \times 10^{-10}$$

$$n = 10.27904829 \Rightarrow 11$$

Thus, we at least need $n+1 = 12$ points.

3.




Code is given on the next page, and by comparing we can find that the cubic spin line method definately gives a much better approximation.

```python
def a_generator(n, i_start, i_end):
    h = (i_end - i_start) / (n-1)
    hs = []
    B = []
    for i in range(1, n + 1):
        hs.append(i_start + (i-1)*h)
        B.append([hs[-1] ** j for j in range(n)])
    B = np.array(B)
    f = np.array([1/(1+25 * x ** 2) for x in hs])
    a = LA.solve(B, f)

    return a

def cubic_denerator(n, i_start, i_end):
    h = (i_end - i_start) / (n-1)
    hs = []
    fx = []
    for i in range(1, n + 1):
        hs.append(i_start + (i-1)*h)
        fx.append(origin(hs[-1]))
    return scipy.interpolate.CubicSpline(hs, fx)

def poly_est(x, a):
    i = 0
    y = 0
    for item in a:
        y += item * x ** i
        i+= 1
    return y

if __name__ == "__main__":
    # n = 11
    a = a_generator(11, -1, 1)

    x_val_11 = []
    original_fun_11 = []
    poly_11 = []

    for i in np.arange(-1, 1, 0.001):
        x_val_11.append(i)
        original_fun_11.append(origin(i))
        poly_11.append(poly_est(i, a))
    cubic_11 = cubic_denerator(11, -1, 1)

    plt.plot(x_val_11, original_fun_11, "r", x_val_11, poly_11, "--b", x_val_11, cubic_11(x_val_11), '--g')
    plt.xlabel("t")
    plt.ylabel("f(t)")
    plt.legend(("f(t)", "poly 11", "cubic 11"))
    plt.show()

    # n = 21
    a = a_generator(21, -1, 1)

    x_val_21 = []
    original_fun_21 = []
    poly_21 = []

    for i in np.arange(-1, 1, 0.001):
        x_val_21.append(i)
        original_fun_21.append(origin(i))
        poly_21.append(poly_est(i, a))
    cubic_21 = cubic_denerator(21, -1, 1)

    plt.plot(x_val_21, original_fun_21, "r", x_val_21, poly_21, "--b", x_val_21, cubic_21(x_val_21), '--g')
    plt.xlabel("t")
    plt.ylabel("f(t)")
    plt.legend(("f(t)", "poly 21", "cubic 21"))
    plt.show()
```