AMSC 460 Homework Set 2

Note:

• Due: 2pm, April 01, 2021. Late submission is subject to automatic 20% reduction.

Due: 21.04.01

 When submitting, make sure you upload your matlab or python codes, in addition to your answer sheets to analytical questions.

Topic: Newton Interpolation with Divided Difference

Consider a polynomial interpolation problem $p_n(x)$ of an (unknown) function f(x),

 $p_n(x) = \sum_{j=0}^{n} c_j \varphi_j(x)$ (1)

where c_i and $\varphi_i(x)$ are the *j*-th interpolating constant and function, respectively. Using (n+1) data sample $\{(x_i, y_i=f(x_i))\}$ for j=0,...,n, Newton interpolation using divided difference can be written as

$$c_j = f[x_0, \dots, x_j]$$

$$\varphi_j(x) = \prod_{i=0}^{j-1} (x - x_i)$$
(2)

where

$$f[x_0, ..., x_j] = \frac{f[x_i] = y_i}{f[x_1, ..., x_j] - f[x_0, ..., x_{j-1}]} \frac{x_j - x_0}{x_j - x_0}$$

Main Problems

- 1. Show that the formula (2) is consistent with the data sample $\{(x_i, y_i)\}$
- 2. Write a pseudocode, as computationally efficient as possible (i.e., minimize number of operations)
- 3. Write a matlab or python code based on the pseudocode
- 4. For a data sample $\{(1,3),(5,11),(2,2),(4,12)\}$,
 - a. Obtain analytical solution, i.e., obtain c_i and $\varphi_i(x)$ by hand
 - b. Show that the analytical solution is consistent with the data sample
 - c. Predict $f(x^*)$ at $x^*=3$, i.e., compute it by hand
- 5. Using the same data sample
 - a. Obtain c_i and $\varphi_i(x)$ by your matlab or python code, verify against your analytical solution obtained in Question 4 above
 - b. Verify that the code reproduces the sample data
 - c. Verify that the code reproduces $f(x^*=3)$ obtained in Question 4 above

Bonus points

B. Derive the formula (2) analytically.

$$C_{3} = \{C_{10}, C_{10}, X_{11}\}$$

$$Q_{1}(X_{1}) = T_{10}^{-1}(X_{10}, X_{11})$$

$$Q_{2}(X_{10}) = \{C_{10}, C_{10}, X_{10}\}$$

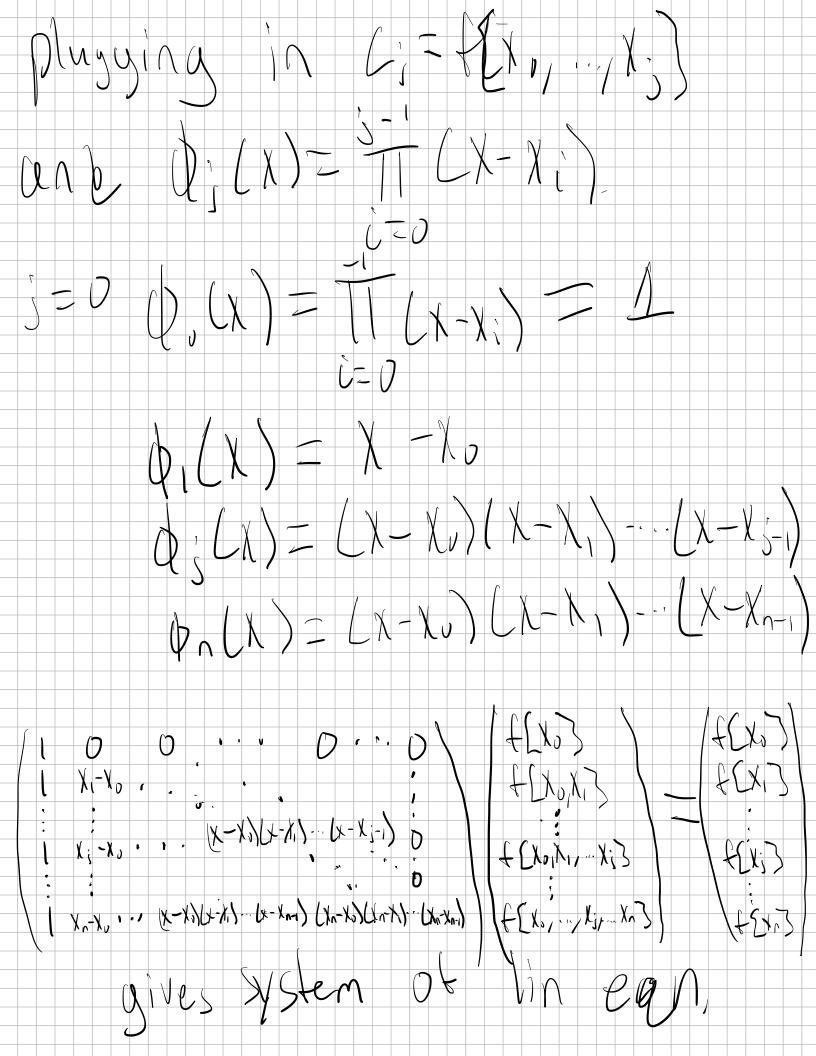
$$Q_{10}(X_{10}) = \{C_{10}, C_{10}, C_{10}, C_{10}, C_{10}\}$$

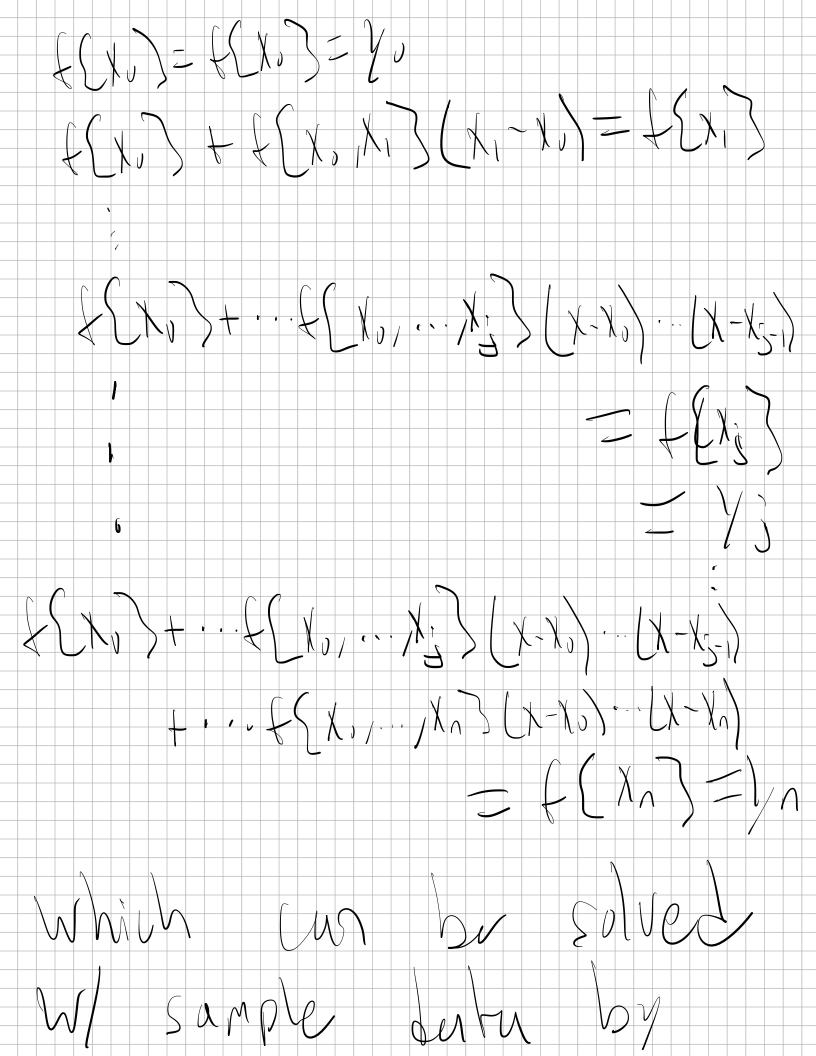
$$Q_{10}(X_{10}) = \{C_{10}, C_{10}, C_{10}, C_{10}, C_{10}\}$$

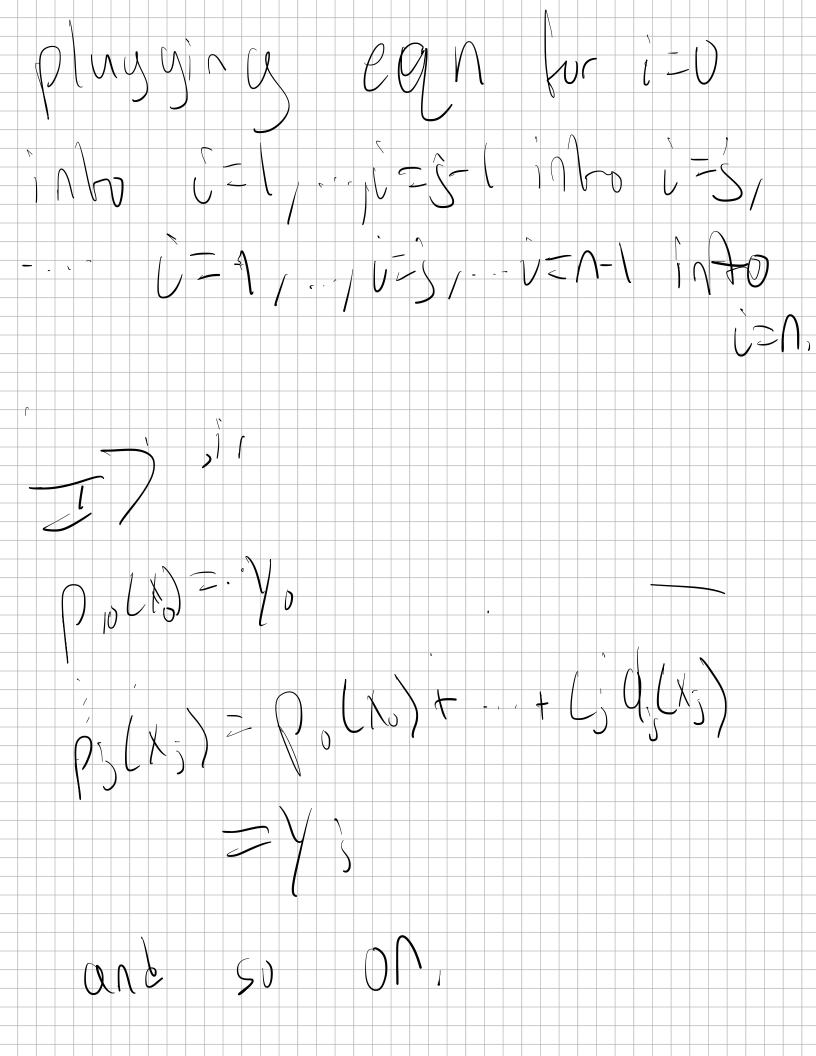
$$Q_{10}(X_{10}) = \{C_{10}, C_{10}, C_{10}, C_{10}, C_{10}, C_{10}\}$$

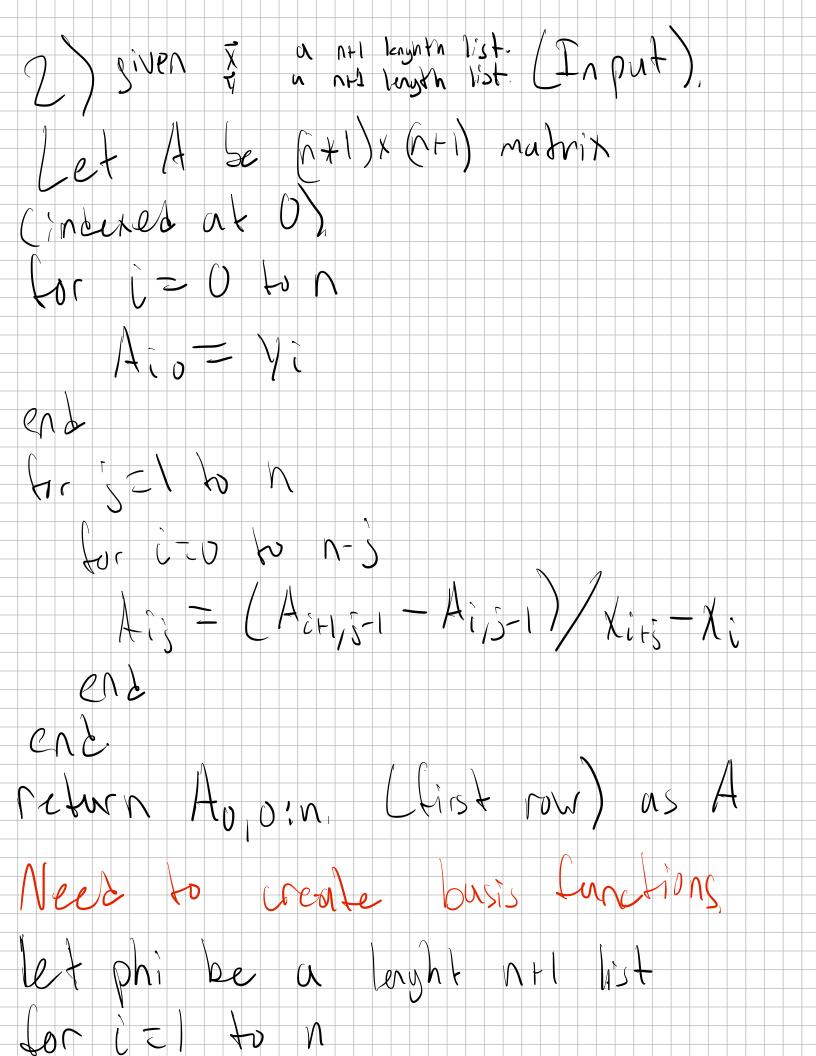
$$Q_{10}(X_{10}) = \{C_{10}, C_{10}, C_{10}, C_{10}, C_{10}, C_{10}, C_{10}, C_{10}\}$$

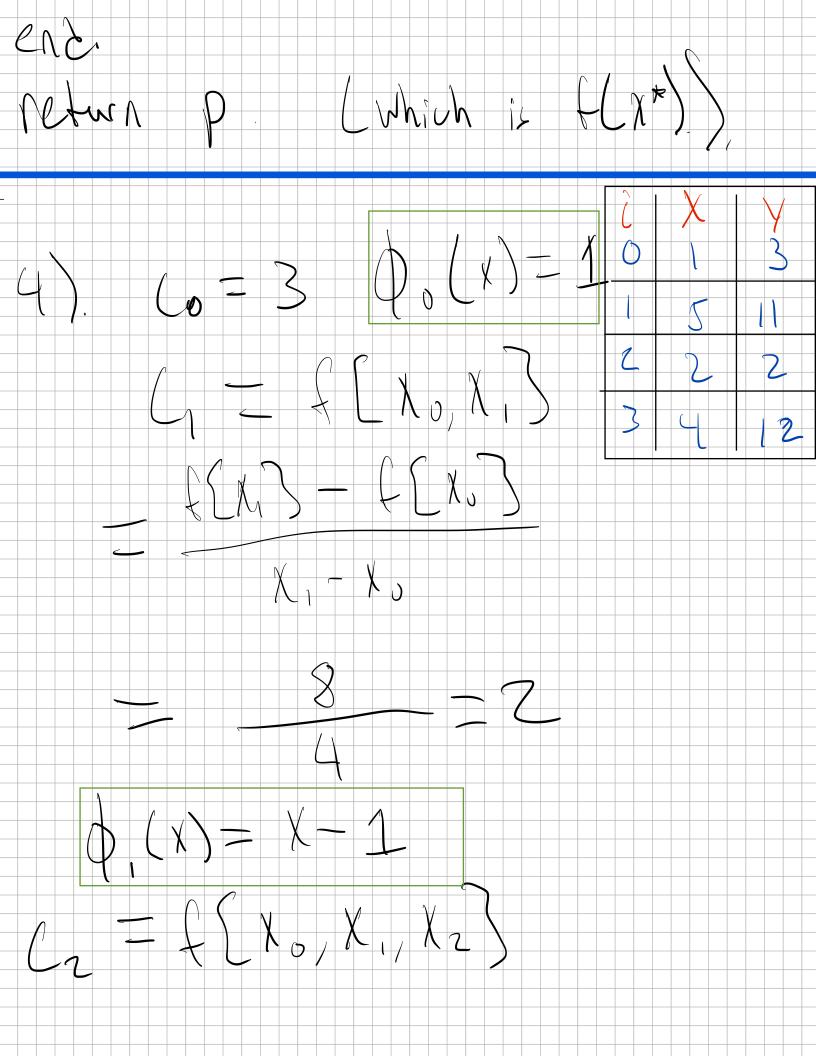
$$Q_{10}(X_{10}) = \{C_{10}, C_{10}, C_{10},$$

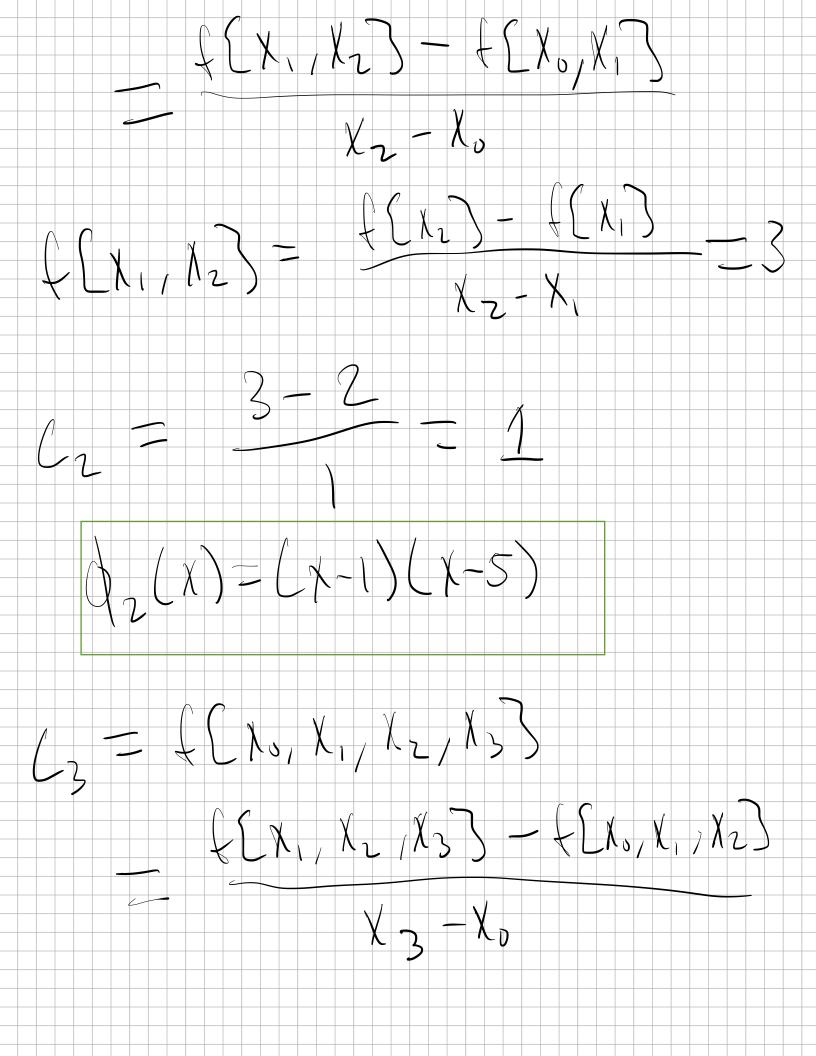






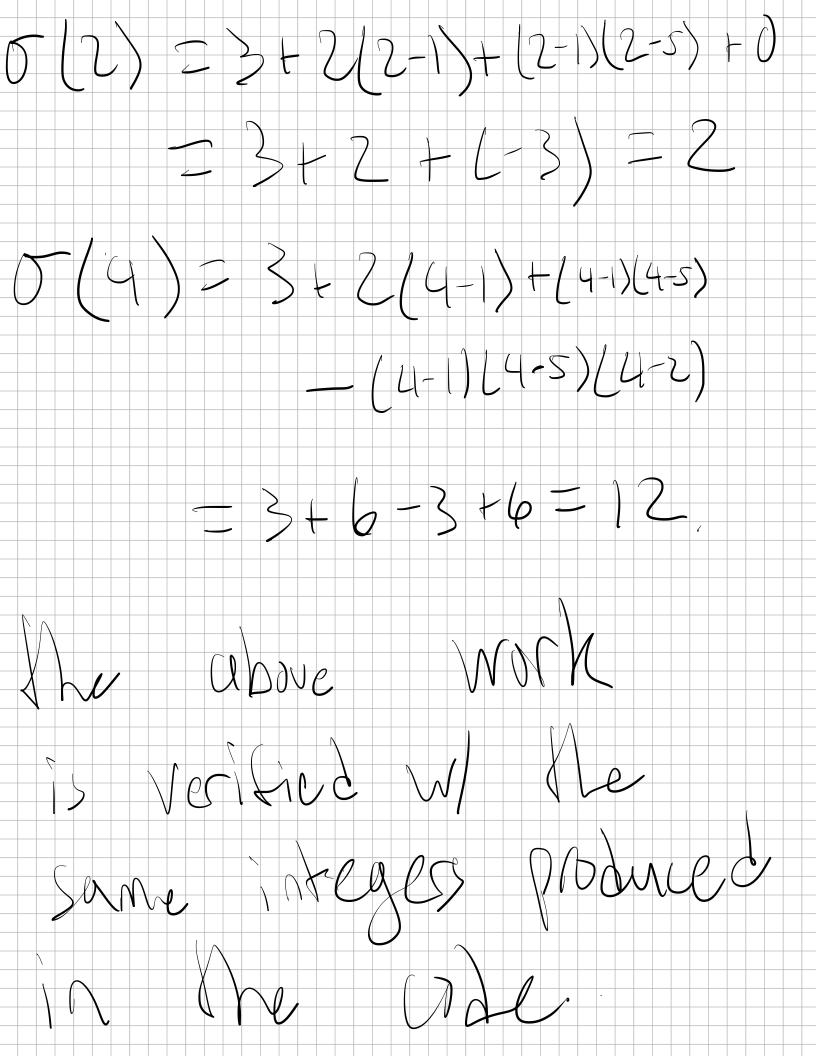






$$\begin{cases} \{X_{1}, X_{2}, X_{3}\} = \{X_{2}, X_{3}\} - \{X_{1}, X_{2}\} \\ \{X_{1}, X_{3}\} = \{X_{2}\} - \{X_{2}\} - \{X_{2}\} = 5 \end{cases}$$

$$\begin{cases} \{X_{1}, X_{2}, X_{3}\} = \{X_{2}\} - \{X_{2$$



HW2

April 1, 2021

```
[1]: import numpy as np
     def get_coeff(x: list, y: list) -> np.ndarray:
         #This will do divided differences
         A = np.zeros((len(x),len(y)), dtype=float)
         for row loc in range(A.shape[0]):
             A[row_loc,0] = y[row_loc]
         for col loc in range(1, A.shape[0]):
             for row_loc in range(A.shape[0] - col_loc):
                 A[row_loc,col_loc] = (A[row_loc+1,col_loc-1] -_
     →A[row_loc,col_loc-1])/(x[row_loc+col_loc] - x[row_loc])
         #coeff are stored along the first row of the array
         return A[0,:]
     def make_phi(x: list) -> np.ndarray:
         phi = np.zeros((len(x),))
         for idx in range(1, len(x)):
             phi[idx] = -x[idx-1]
         #first basis function for a polynomial is always degree 0
         phi[0] = 1
         return phi
     def interpolate x star(coeff: np.ndarray, phi: np.ndarray, x star):
         phi[1:] = phi[1:] + x_star
         for basis in range(1, phi.shape[0]):
             phi[basis] = phi[basis]*phi[basis-1]
         sol = coeff.dot(phi)
         print("phi is: " + str(phi))
         print("coeff are: " + str(coeff))
         return sol
[2]: def solve_newton_interpolation(x: list, y: list, x_star: int) -> float:
         return interpolate_x_star(get_coeff(x,y), make_phi(x), x_star)
[3]: x = [1, 5, 2, 4]
     y = [3, 11, 2, 12]
     x_star = 3
```

```
ans = solve_newton_interpolation(x, y, 3)
    print("f("+str(x_star)+")=" + str(ans) + "\n")
   phi is: [ 1. 2. -4. -4.]
   coeff are: [ 3. 2. 1. -1.]
   f(3)=7.0
[4]: for x_star in x:
       \hookrightarrow x_star))+"\n")
   phi is: [ 1. 0. -0. 0.]
   coeff are: [ 3. 2. 1. -1.]
   f(1)=3.0
   phi is: [1. 4. 0. 0.]
   coeff are: [ 3. 2. 1. -1.]
   f(5)=11.0
   phi is: [ 1. 1. -3. -0.]
   coeff are: [ 3. 2. 1. -1.]
   f(2)=2.0
   phi is: [ 1. 3. -3. -6.]
   coeff are: [ 3. 2. 1. -1.]
   f(4)=12.0
[]:
```