Question 3:

Date..........................

3\. (a) $F_n(t) = \cos(n\cos^{-1}(t))$

$F_0(t) = \cos(0 \times \cos^{-1}(t))$
$= \cos(0)$

$\Rightarrow \boxed{F_0(t) = 1}$ —— (i)

$F_1(t) = \cos(1 \times \cos^{-1}(t))$

$= \cos(\cos^{-1} t)$

$\Rightarrow \boxed{F_1(t) = t}$ —— (ii)

for $F_{n+1}(t) = \cos((n+1)\cos^{-1} t)$

$= \cos(n\cos^{-1}t + \cos^{-1}t)$

$= \cos(n\cos^{-1}t)\cos(\cos^{-1}t) - \sin(n\cos^{-1}t)\cdot\sin(\cos^{-1}t)$ —— (iv)

$F_{n-1}(t) = \cos((n-1)\cos^{-1}t)$

$= \cos(n\cos^{-1}t - \cos^{-1}t)$

$= \cos(n\cos^{-1}t)\cdot\cos(\cos^{-1}t) + \sin(n\cos^{-1}t)\cdot\sin(\cos^{-1}t)$ (v)

adding (iv) & (v)

$\Rightarrow F_{n+1}(t) + F_{n-1}(t) = 2\cos(n\cos^{-1}t)\cdot\cos(\cos^{-1}t)$

$\Rightarrow \boxed{F_{n+1}(t) + F_{n-1}(t) = 2t\,F_n(t)}$ —— (vi)

Spiral

Teacher's Sign ....

(b) Since $F_n(t)$ satisfy all the three conditions of Chebyshev polynomial. (By eq (i), (ii) & (vi))

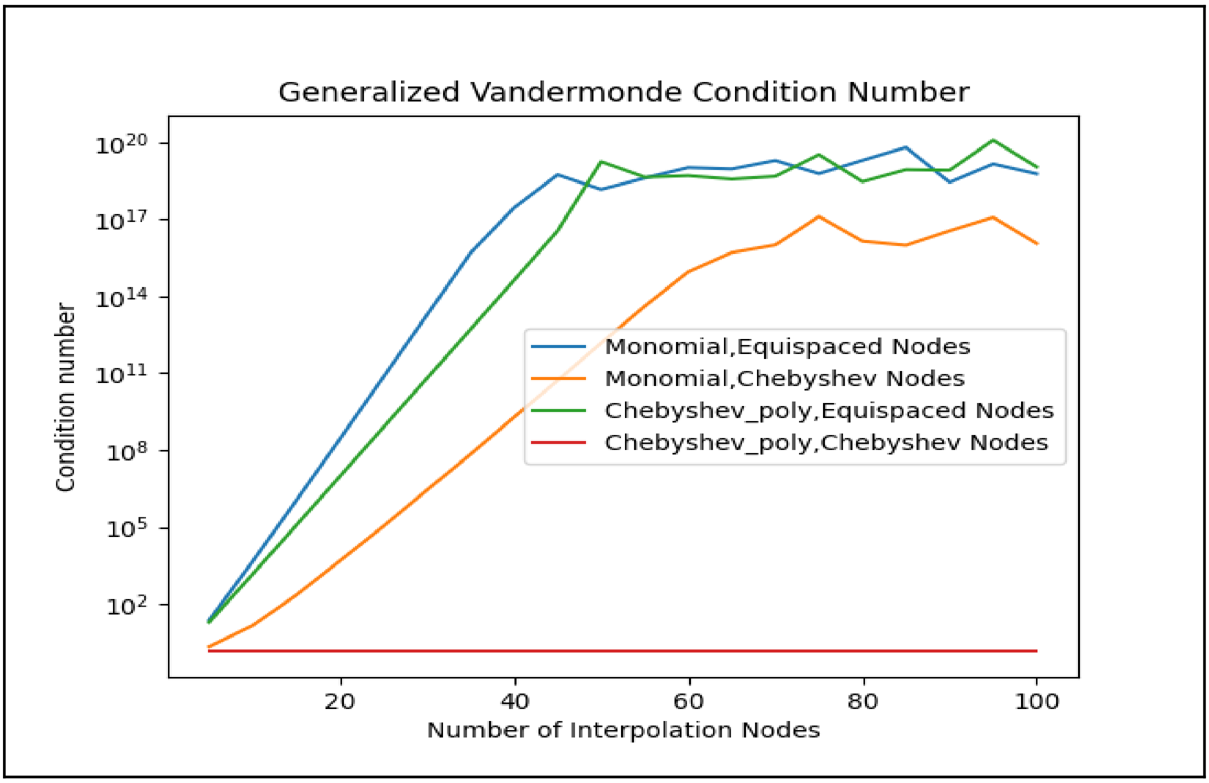
Hence $F_n(t)$ is Chebyshev polynomial.

source code:

```
import numpy as np
import numpy.linalg as la
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
def equispaced(n):
    return np.linspace(-1, 1, n)
def chebyshev(n):
    X = np.zeros(n)
    for i in range(0, n):
        X[i] = np.cos((2*i + 1)/(2*n) * np.pi)
    return X
def vandermonde(X, m):
    def chebyshev_poly(n, x):
        return np.cos(n * np.arccos(x))

    def monomial(n, x):
        return x**n
    n = X.size
    V_mono = np.zeros((n, m))
    V_cheb = np.zeros((n, m))
    for i in range(n):
        for j in range(m):
            V_mono[i, j] = monomial(j, X[i])
            V_cheb[i, j] = chebyshev_poly(j, X[i])
    return V_mono, V_cheb

values_mono_equi = []
values_mono_cheb = []
values_cheb_equi = []
values_cheb_cheb = []
for n in range(5, 101, 5):
    X_equi = equispaced(n)
    X_cheb = chebyshev(n)

    mono_equi, cheb_equi = vandermonde(X_equi, n)
    mono_cheb, cheb_cheb = vandermonde(X_cheb, n)
```

```
    values_mono_equi.append([n, la.cond(mono_equi)])
    values_cheb_equi.append([n, la.cond(cheb_equi)])
    values_mono_cheb.append([n, la.cond(mono_cheb)])
    values_cheb_cheb.append([n, la.cond(cheb_cheb)])

values_mono_equi = np.array(values_mono_equi)
values_cheb_equi = np.array(values_cheb_equi)
values_mono_cheb = np.array(values_mono_cheb)
values_cheb_cheb = np.array(values_cheb_cheb)

plt.xlabel("Number of Interpolation Nodes")
plt.ylabel('Condition number')
plt.title("Generalized Vandermonde Condition Number")

plt.semilogy(values_mono_equi[:, 0], values_mono_equi[:, 1], label='Monomial,Equispaced
Nodes')
plt.semilogy(values_cheb_equi[:, 0], values_cheb_equi[:, 1], label='Monomial,Chebyshev
Nodes')
plt.semilogy(values_mono_cheb[:, 0], values_mono_cheb[:, 1],
label='Chebyshev_poly,Equispaced Nodes')
plt.semilogy(values_cheb_cheb[:, 0], values_cheb_cheb[:, 1],
label='Chebyshev_poly,Chebyshev Nodes')

plt.legend()
plt.savefig("problem_3.png")
plt.show()
```

Plot:



d) By looking at the plot, We can clearly say that the combination of Chebyshev polynomial with Chebyshev nodes performs best. And it has a constant condition number for given number of Interpolation nodes.

Question 4:

Date........................

Suppose $p_k(t)$ interpolating f on $t_1, t_2 \dots t_k$

$$\therefore p_k(t) = a_1 + a_2(t-t_1) + a_3(t-t_1)(t-t_2) + \dots + a_k(t-t_1)\cdots(t-t_{k-1})$$

Where $a_1, a_2 \dots a_{k-1}$ are known by induction hypothesis and given by the divided difference.

Since we know that

$$f(t_k) = p_k(t)$$

therefore

$$f(t_k) = a_1 + a_2(t_k - t_1) + a_3(t_k - t_1)(t_k - t_2) + \dots + a_k(t_k - t_1)\dots(t_k - t_{k-1})$$

$$\frac{f(t_k) - a_1}{t_k - t_1} = a_2 + a_3(t_k - t_2) + a_4(t_k - t_2)(t_k - t_3) + \dots + a_k(t_k - t_2)\cdots(t_k - t_{k-1})$$

Since $a_1 = f(t_1)$ (By induction hypothesis)

now substituting the value of a_1.

we have

$$\frac{f(t_k) - f(t_1)}{t_k - t_1} = a_2 + a_3(t_k - t_2) + \dots$$

$$\Rightarrow f[t_1, t_2] = a_2 + a_3(t_k - t_2) + \dots + a_k(t_k - t_2)\cdots(t_k - t_{k-1})$$

Spiral

Teacher's Sign ....

now proceeding with the same steps again & again we have

$$\frac{f[t_1, t_2, \ldots t_{k-1}] - a_{k-1}}{(t_k - t_{k-1})} = a_k$$

$$\Rightarrow \quad f[t_1, t_2, \ldots t_k] = a_k$$

Hence proved

(b) Given:- Three data points $(-1,1), (0,0), (1,1)$

find:- the interpolating polynomial of degree 2. using (i) monomial (ii) Lagrange (iii) Newton basis.

Solution: By using Monomial:-

Monomial Basis:- Linear system is given by

$$Ax = \begin{bmatrix} 1 & t_1 & t_1^2 \\ 1 & t_2 & t_2^2 \\ 1 & t_3 & t_3^2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = y$$

$$\Rightarrow \quad \begin{bmatrix} 1 & -1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

by solving using gaussian elimination

Date..........................

$$\left[\begin{array}{ccc|c} 1 & -1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{array}\right] \xrightarrow[R_3 \to R_3 - R_2]{R_1 \to R_1 - R_2} \left[\begin{array}{ccc|c} 0 & -1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{array}\right]$$

$$\Big\downarrow R_3 \to R_3 + R_1$$

$$\left[\begin{array}{ccc|c} 0 & -1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 2 & 2 \end{array}\right]$$

$\Rightarrow \quad x_1 = 0 \quad , x_3 = 1$

$-x_2 + x_3 = 1$

$\Rightarrow \quad -x_2 + \cancel{1} = \cancel{1}$

$\Rightarrow \quad x_2 = 0$

$$\Rightarrow \quad x = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$\therefore$ Polynomial is given as

$$P_2(t) = 0 \cdot + 0 \cdot t + 1 \cdot t^2$$

$$\Rightarrow \boxed{P_2(t) = t^2} \quad \text{---} \quad (i)$$

By Using Lagrange Basis:-

Lagrange basis function are given by

$$l_j(t) = l(t) \frac{w_j}{t - t_j} \quad , j = 1, 2, 3$$

Spiral

Teacher's Sign ....

Date..........................

where $l(t) = (t-t_1)(t-t_2)(t-t_3)$

& $w_j = \dfrac{1}{l'(t_j)} \quad \forall\ j = 1, 2, 3$

$\Rightarrow \quad w_1 = \dfrac{1}{(t_1-t_2)(t_1-t_3)}$

$w_2 = \dfrac{1}{(t_2-t_1)(t_2-t_3)}$

$w_3 = \dfrac{1}{(t_3-t_1)(t_3-t_2)}$

$\therefore$ polynomial is given by

$$P_2(t) = l(t)\left[y_1 \frac{w_2}{t-t_1} + y_2 \frac{w_2}{t-t_2} + y_3 \frac{w_3}{t-t_3}\right]$$

now substituting the value of all the points in the above equation

$\Rightarrow \quad l(t) = (t+1)(t-0)(t-1)$
$= t(t+1)(t-1)$

$\Rightarrow \boxed{l(t) = t^3 - t}$

$w_1 = \dfrac{1}{(-1-0)(-1-1)} = \dfrac{1}{2}$

$w_2 = \dfrac{1}{(0+1)(0-1)} = -1$

Date.........................

$$w_3 = \frac{1}{(1+1)(1-0)} = \frac{1}{2}$$

$$\therefore P_2(t) = (t^3-t)\left[1 \cdot \frac{1/2}{t+1} + 0 \cdot \frac{(-1)}{t-0} + 1 \cdot \frac{1/2}{t-1}\right]$$

$$= (t^3-t)\left[\frac{1}{2(t+1)} + \frac{1}{2(t-1)}\right]$$

$$= (t^3-t)\left[\frac{t-\cancel{1}+t+\cancel{1}}{2(t^2-1)}\right]$$

$$= t\cancel{(t^2-1)} \times \frac{\cancel{2}t}{\cancel{2(t^2-1)}}$$

$$\Rightarrow \boxed{P_2(t) = t^2} \quad \text{---} \quad (ii)$$

By Using Newton Basis :-

Using Newton Basis, linear System is given as

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & t_2-t_1 & 0 \\ 1 & t_3-t_1 & (t_3-t_1)(t_3-t_2) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

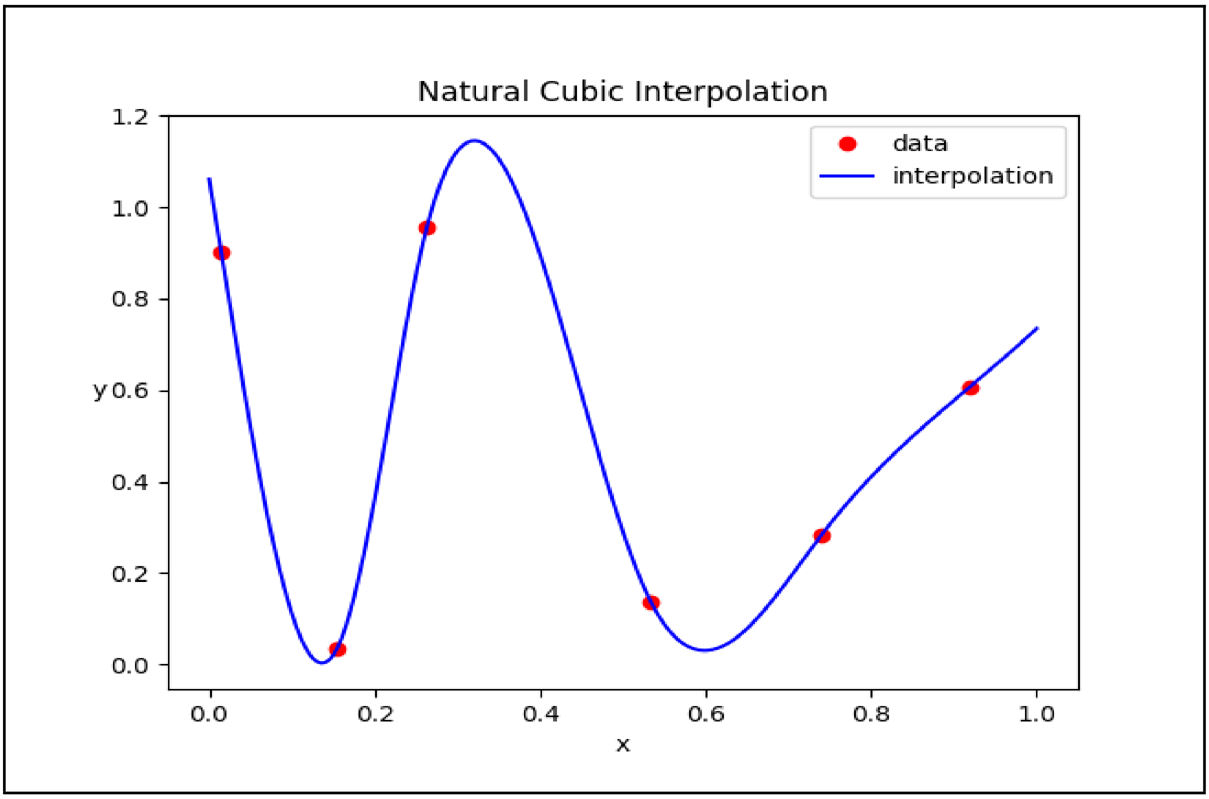
$\therefore$ for the given points, our linear system will become as

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 2 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

(c)
Source Code

```
import random
import numpy as np
import matplotlib.pyplot as plt
from numpy.lib.function_base import rot90
from scipy.interpolate import CubicSpline
np.random.seed(12)

x = np.random.rand(6)
y = np.random.rand(6)
x = np.sort(x)

cubic_spline = CubicSpline(x, y, bc_type='natural', extrapolate=True)

plt.plot(x, y, 'o', label='data', color='r')
plt.plot(np.linspace(0, 1-1e-20, 200), cubic_spline(np.linspace(0, 1-1e-20, 200)),
label="interpolation", color='b')

plt.legend(loc='best')
plt.title('Natural Cubic Interpolation')
plt.xlabel('x')
plt.ylabel('y', rotation=0)
plt.savefig('problem_4c.png')
plt.show()
```

Plot:



Question 1:

Date..........

1. (a) $x_{k+1} = x_k - \frac{f(x_k)}{d}$

Suppose $g(x_k) = x_k - \frac{f(x_k)}{f'(x_k)}$

$$g'(x_k) = 1 - \frac{f'(x_k)^2 - f(x_k) \cdot f''(x_k)}{f'(x_k)^2}$$

$$g'(x_k) = \frac{\cancel{f'(x_k)^2} - \cancel{f'(x_k)^2} + f(x_k) \cdot f''(x_k)}{f'(x_k)^2}$$

$$\Rightarrow g'(x_k) = \frac{f(x_k) \cdot f''(x_k)}{f'(x_k)^2}$$

for the convergence of newton method

$$|g'(x_k)| < 1$$

$$\Rightarrow |f(x_k) \cdot f''(x_k)| < d^2$$

$$\Rightarrow \boxed{d^2 > |f(x_k) \cdot f''(x_k)|}$$

(b). Convergence rate,

$$E_{k+1} = \frac{f''(x_0)}{2f'(x_0)} E_k^2$$

(c) for quadratic convergence, $d \neq 0$

Spiral

Teacher's Sign ....

b)
Source code:

```
import numpy as np
def newton(func, der_func, x_0, epsilon):
    x_n = x_0
    while func(x_n) > epsilon:
        y = func(x_n)
        y_ = der_func(x_n)
        x_n = x_n-y/y_
    return x_n


def func1(x): return x**2 - 1
def der_func1(x): return 2*x


def func2(x): return (x - 1)**4
def der_func2(x): return 4*(x - 1)**3


def func3(x): return x - np.cos(x)
def der_func3(x): return 1 + np.sin(x)


print(newton(func1, der_func1, 10**6, 1e-100))
print(newton(func2, der_func2, 10, 1e-60))
print(newton(func3, der_func3, 1, 1e-100))
```

Output

```
1.0
1.0000000000000009
0.7390851332151607
```