In [ ]:
```python
import numpy as np
import matplotlib.pyplot as plt
```

In [ ]:
```python
def f(x):
    return np.exp(-0.3*x)

def get_Interpolating_pts(n: int):
    xs = np.zeros(n)
    for k in range(n):
        xs[k] = np.cos((2*k+1)/(2*n)*np.pi)

    return xs

def compute_weights(xs : np.ndarray):
    #Calculate Big W
    n = xs.size
    W = np.zeros((n, n))
    for i in range(n):
        for j in range(n):
            if i != j: W[i][j] = 1.0/(xs[j] - xs[i])
    # Calc the weights
    ws = np.ones(n)
    for i in range(n):
        for j in range(n):
            if i != j: ws[i]*= W[i][j]

    return ws

def get_approximation(ws: np.ndarray, fs: np.ndarray, xs: np.ndarray, Cheb_nodes: np.ndarray):
    poly_approx = np.zeros_like(xs)
    n = ws.size
    for i, x in enumerate(xs):
        #Get numerator
        num = 0
        for j in range(n):
            num += (ws[j] * fs[j]) / (x - Cheb_nodes[j])
        #Get Denominator
        den = 0
        for j in range(n):
            den += (ws[j]) / (x - Cheb_nodes[j])

        #Divide
        poly_approx[i] = num / den

    return poly_approx
```

In [ ]:
```python
#Set interpolating points
num_nodes = 5
Cheb_nodes = get_Interpolating_pts(num_nodes)

#Compute weights
ws = compute_weights(Cheb_nodes)

# Get function values
fs = f(Cheb_nodes)

# Get polynomial approximation
xs = np.linspace(1, -1, 500)
poly_approx = get_approximation(ws, fs, xs, Cheb_nodes)
f_true = f(xs)
```

In [ ]:
```python
fig, axs = plt.subplots(1, 1, figsize = (8, 8))
f_nodes = np.zeros(Cheb_nodes.size)

for j in range(Cheb_nodes.size):
    for pt in poly_approx:
        if abs(pt - Cheb_nodes[j]) < 1e-6:
            f_nodes[j] = abs(pt - f(Cheb_nodes[j]))
            break


axs.plot(xs, np.abs(poly_approx - f_true))
for i, f_node in enumerate(f_nodes):
    axs.scatter(Cheb_nodes[i], f_node, c = 'k')
    axs.annotate(f"$x_{4-i}$", xy=(Cheb_nodes[i] + 1/24, f_node - 2e-8))
axs.grid()
fig.gca().set_facecolor((0.9, 0.9, 0.9))
axs.set_title("Absolute Error for $f(x) = e^{-0.3x}$")
axs.set_ylabel("Error")
axs.set_axisbelow(True)
```