Homework7

Friday, October 20, 2023 9:52 AM

Tony Samour
APPM 4600

1.) 1. In this problem we find the the polynomial $p(x) = c_n + c_{n-1}x + c_{n-3}x^2 + \cdots + c_1 x^{n-1}$ that interpolates the data $(x_j, y_j) = (x_j, f(x_j)), j = 1, \ldots, n$.

(a) Assume $(x_j, y_j), j = 1, \ldots, n$ are given. Derive the system $V\mathbf{c} = \mathbf{y}$ that determines the coefficients $\mathbf{c} = [c_1, \ldots, c_n]^T$ (here $\mathbf{y} = [y_1, y_2, \ldots, y_n]^T$), that is, find how the matrix V looks like.

To solve the system of equations you can simply use inversion from Numpy's linear algebra package. You will write your polynomial evaluator.

$$p(x_j) = c_n + c_{n-1}x_j + c_{n-2}x_j^2 + \cdots + c_1 x_j^{n-1} \approx f(x_j), \quad j = 1, \ldots, n$$

$$Vc = y \Rightarrow V^{(n \times n)} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \Rightarrow \begin{bmatrix} x_1^{n-1} & x_1^{n-2} & \cdots & x_1^0 \\ x_2^{n-1} & \cdots & \cdots & \vdots \\ \vdots & \cdots & \cdots & \vdots \\ x_j^{n-1} & \cdots & \cdots & x_j^0 \end{bmatrix} \cdot \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

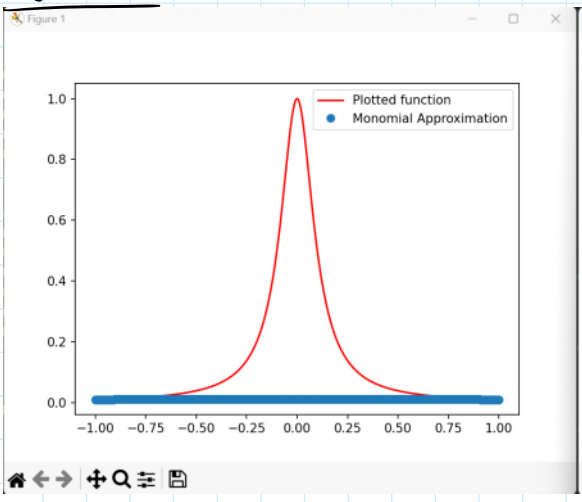
$Vc = y \Rightarrow c = V^{-1}y$, use Python to solve

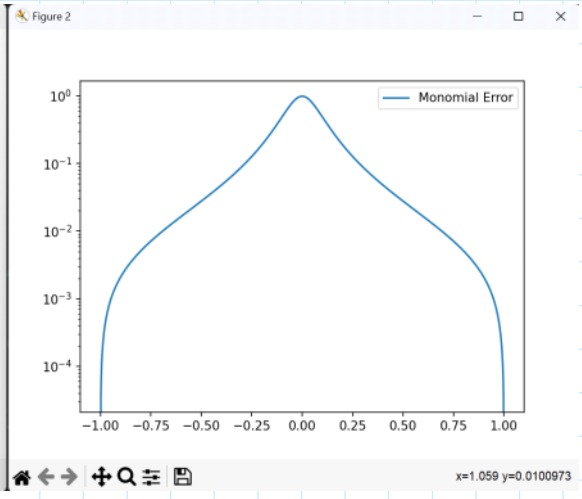
(b) Code

```
def eval_polynomial(xint, xeval, yint, N):
    V = np.zeros([N+1,N+1])
    for r in range (N+1):
        for c in range (N+1):
            V[r][c] = xint[r]**c

    ceval = inv(V).dot(yint)
    poly = []

    for i in xeval:
        n = 0
        for j in range(N+1):
            n += ceval[j] * i**j

        poly.append(n)

    return (ceval, poly)
```

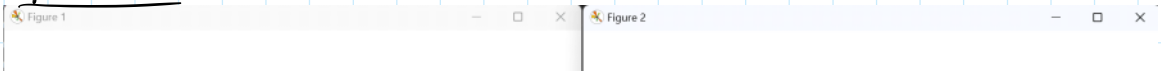
Output Plots:

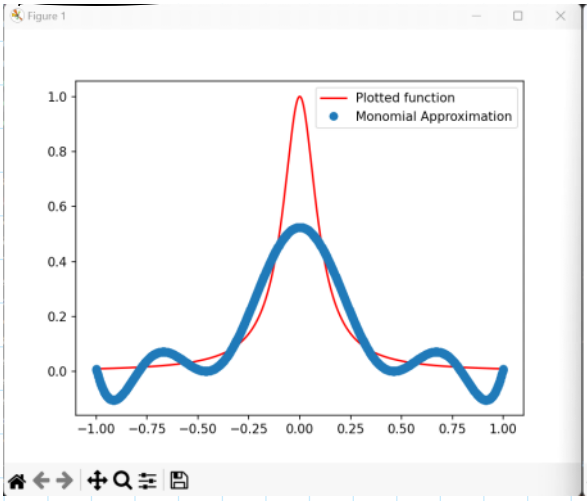
N = 2

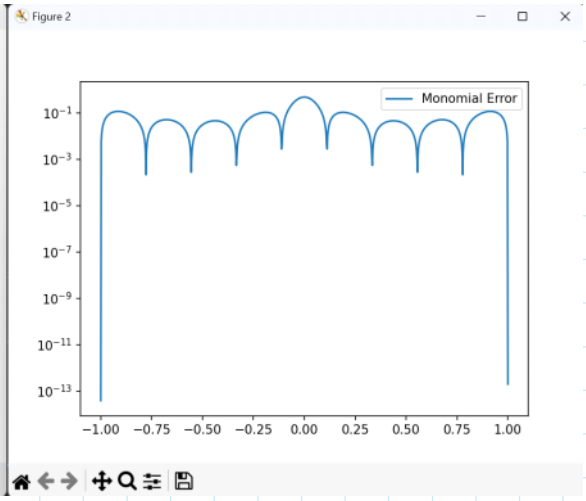




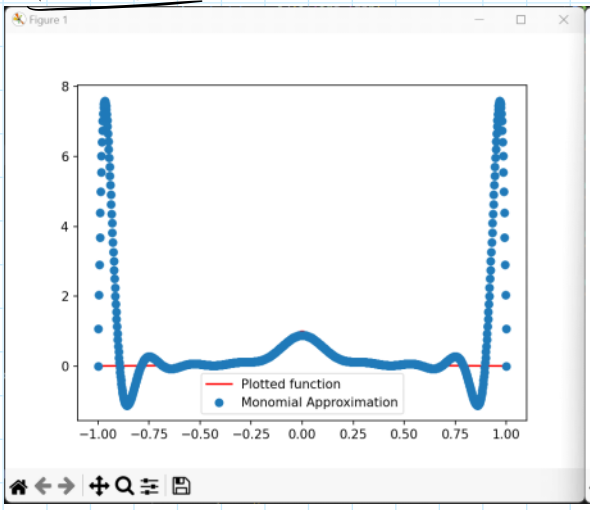
N = 10

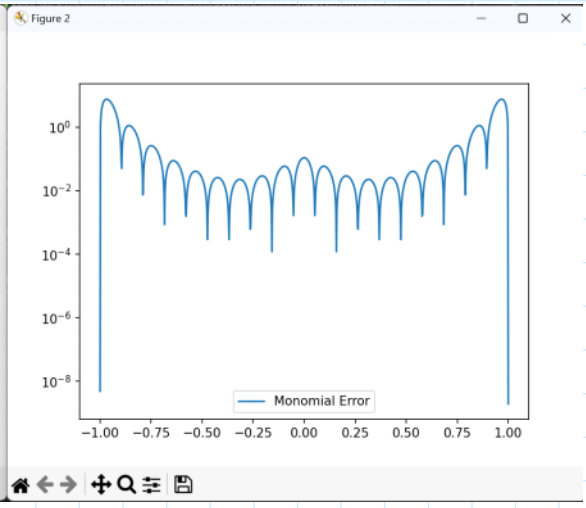






N = 20





As N increases, the approximation becomes more accurate, but the error is still very large near the end points.

2.) 2. Solving the interpolation problem using the monomial basis (as above) is notoriously ill-conditioned, in fact it is possible to show $\texttt{cond(V)} \sim \pi^{-1} e^{\pi/4} (3.1)^n$. A better way of interpolating is to use either of the barycentric Lagrange interpolation formulas:

$$p(x) = \Phi_n(x) \sum_{j=0}^{n} \frac{w_j}{x - x_j} f(x_j).$$

$$p(x) = \frac{\sum_{j=0}^{n} \frac{w_j}{x - x_j} f(x_j)}{\sum_{j=0}^{n} \frac{w_j}{x - x_j}}, \quad x \neq x_j.$$

Where

$$\Phi_n(x) = \prod_{i=0}^{n} (x - x_i), \quad w_j = \frac{1}{\prod_{i=0, i \neq j}^{n} (x_j - x_i)}.$$

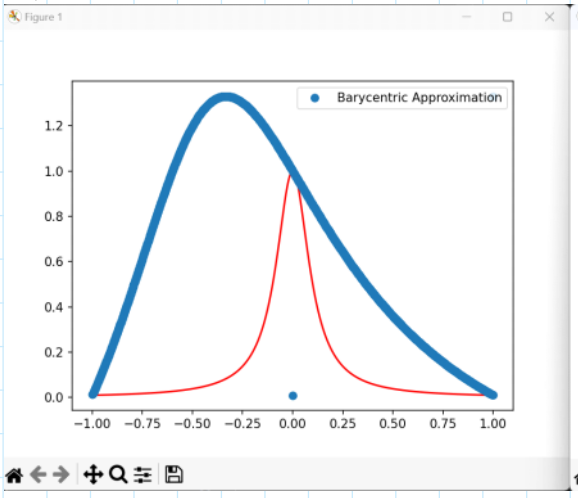
Using either of the above formulas try again to interpolate $f(x)$. Show with some pictures that you still get the same bad behavior close to the endpoints (this is a property of the function $f(x)$ and the distribution of the grid points not of the form of interpolation) but that the approximation is well behaved for small x for very large n.

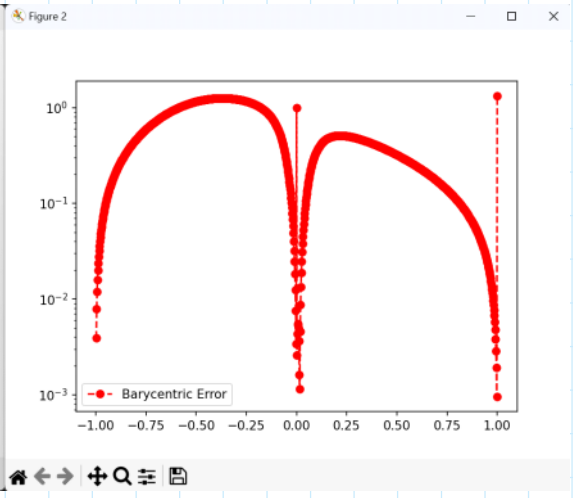
Code:

```python
def eval_barycentric(xeval,xint,yint,N):

    wj = np.ones(N+1)

    for count in range(N+1):
        for jj in range(N+1):
            if (jj != count):
                wj[count] = 1/(wj[count]*(xint[jj] - xint[count]))

    numerator = 0
    denominator = 0

    for i in range(N+1):
        if (xeval != xint[i]):
            numerator += ((wj[i]*yint[i])/(xeval - xint[i]))
            denominator += (wj[i]/(xeval - xint[i]))

    yeval = numerator / denominator

    return yeval
```

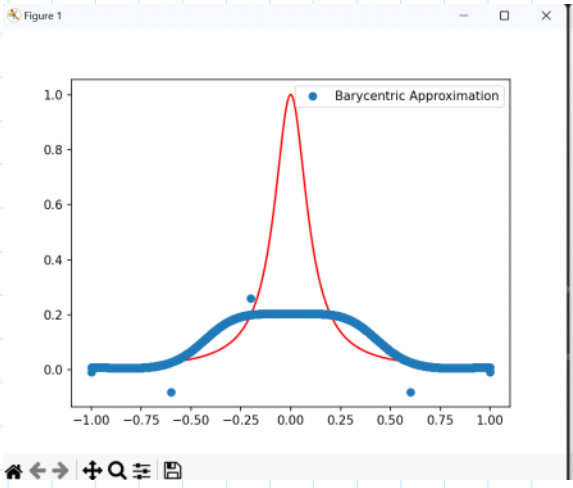
Output:

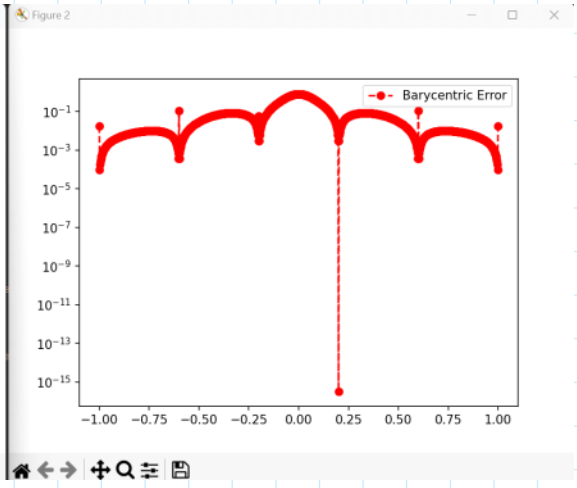
N = 2



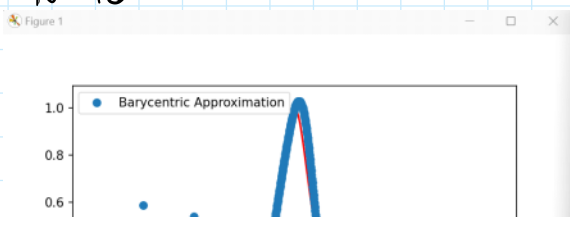


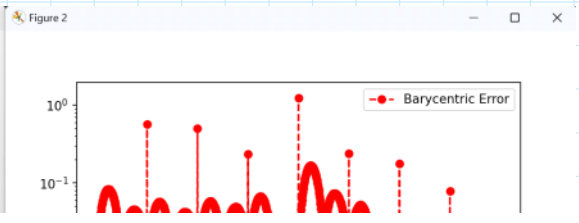
N = 5

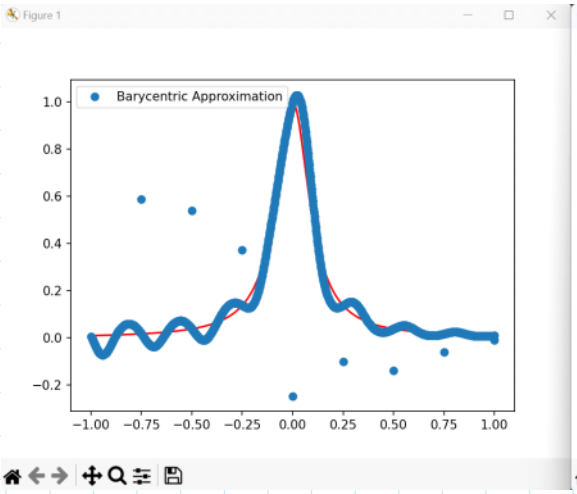


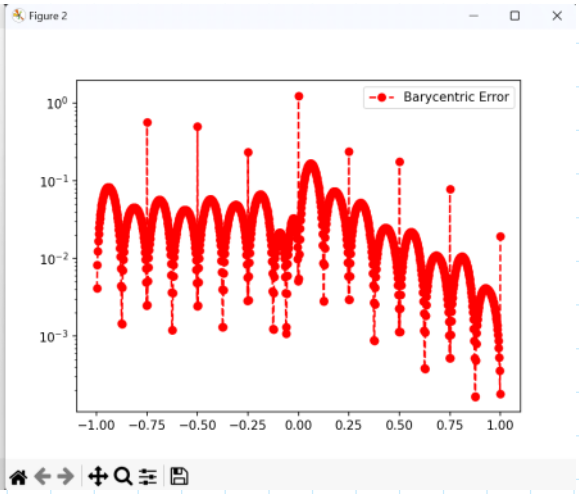


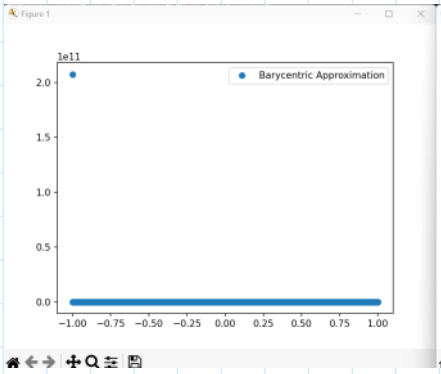
N = 16

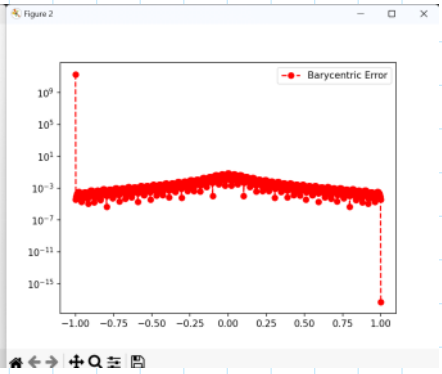












Using the value of $N = 50$, it becomes clear that the endpoints do not behave nicely.

3.)

3. It is much better to interpolate on a grid made up of points that are clustered towards the endpoints. Try to interpolate $f(x)$ in the Chebyshev points

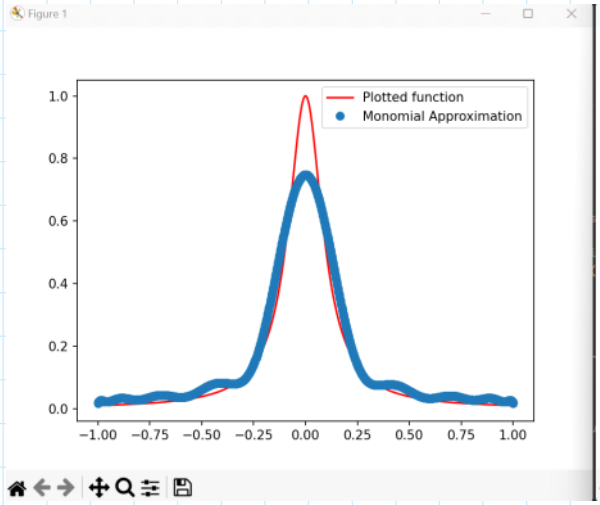
$$x_j = \cos\frac{(2j-1)\pi}{2N}, \quad i = 1, \ldots, N,$$

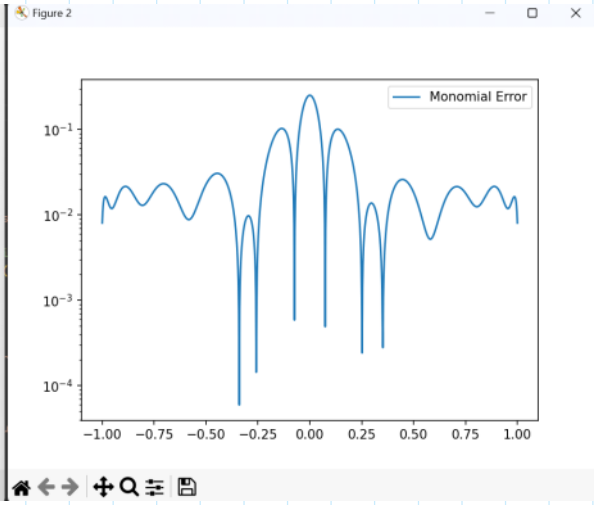
using either of the methods above. Can you get the interpolation to fail now?

Code for Chebyshev interpolation points:

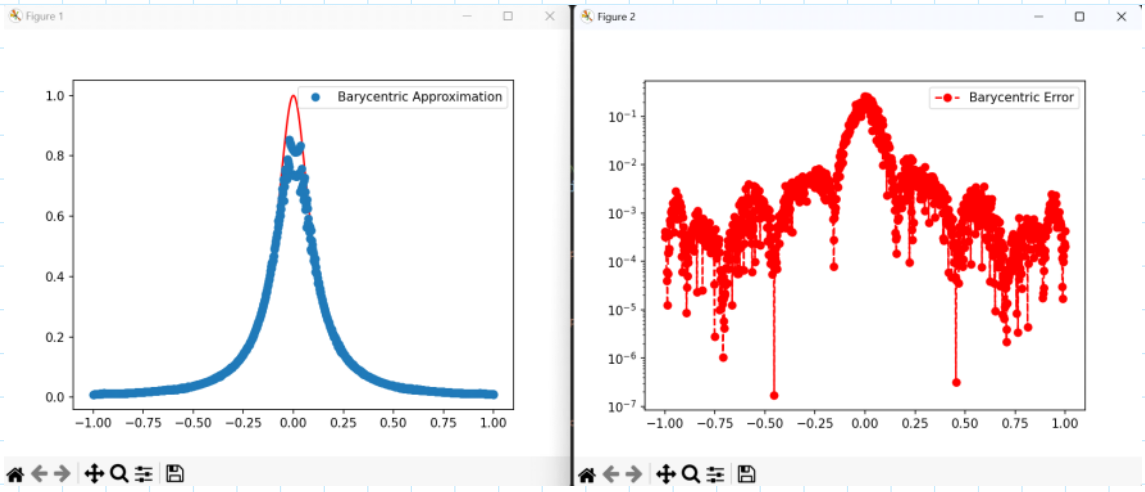
```
xint = np.array([np.cos(((2*j-1)*np.pi)/(2*N)) for j in range(1,N+2)])
```

Output





Using the monomial approximation with $N = 20$. The error near the end points is no longer apparent.



Using a Barycentric approximation with $N = 20$, I got the following plots.

Note: The interpolation fails with large N values and values of N that make the matrix singular for the monomial approximation