16811 A2

September 2022

1.

(a) The implementation is in the q1.py. Run *python q1.py* to get the interpolation result.

(b) By running *python q1.py*, I got the interpolate result for $sin(2\pi x)$ at $x = \frac{1}{10}$ as 0.5877882222892551.

(c) By running *python q1.py*, I got the estimate results:
at n = 2, 0.992062;
at n = 4, 0.9629441579018866;
at n = 40, 0.7157541032066309.
The actual value of $f(0.09)$ is 0.7158708568974157.

(d) the interpolation error for n = 2 is 0.7371572873781101
the interpolation error for n = 4 is 0.493574463844801 47
the interpolation error for n = 6 is 0.7713473670375813
the interpolation error for n = 8 is 1.4928508356212091
the interpolation error for n = 10 is 3.1707382021179353
the interpolation error for n = 12 is 7.074864188499847
the interpolation error for n = 14 is 16.266037632336634
the interpolation error for n = 16 is 38.1467530538924
the interpolation error for n = 18 is 90.72902210700984
the interpolation error for n = 20 is 218.20182193178275
the interpolation error for n = 40 is 1891744.864270032
The error estimate results show that the error increases as n increases. This demonstrates that larger n does not necessary leads to a better result. This make sense, because the accuracy of polynomial interpolation depends on how close the interpolated point is to the middle of the x values of the set of points used. In the case for example we are interpolating $f(x)$ where $x \in [-1, 1]$, as n increases, the middle becomes farther from the interpolated points, and thus the error gets larger.

2.

$$f^{'}(x) = 2\pi cos(2\pi x); f^{''}(x) = -(2\pi)^2 sin(2\pi x); f^{'''}(x) = -(2\pi)^3 cos(2\pi x)$$

1. linear interpolation

$$e_1(x) = \frac{f^{(2)}(\xi)}{2!}(x - x_{i-1})(x - x_i)$$
$$f^{''}(x) = -(2\pi)^2 sin(2\pi x) \Rightarrow \max_{0\leq\xi\leq 1}|f^{''}(\xi)| = (2\pi)^2$$
$$\max_{0\leq x\leq 1}(x - x_{i-1})(x - x_i) = \max_{0\leq y\leq h}|y(y-h)| = \frac{h^2}{4}$$
$$\max_{0\leq x\leq 1}|e_1(x)| \leq \frac{1}{2}\cdot(2\pi)^2\cdot\frac{h^2}{4} = \frac{(\pi)^2 h^2}{2}$$
$$\text{want } \frac{(\pi)^2 h^2}{2} \leq 5\cdot 10^{-7} \Rightarrow h \leq 3.18309886104$$
$$\Rightarrow N = \frac{1}{h} \approxeq 3141.6$$

Therefore, need 3143 data points (or 3142 intervals).

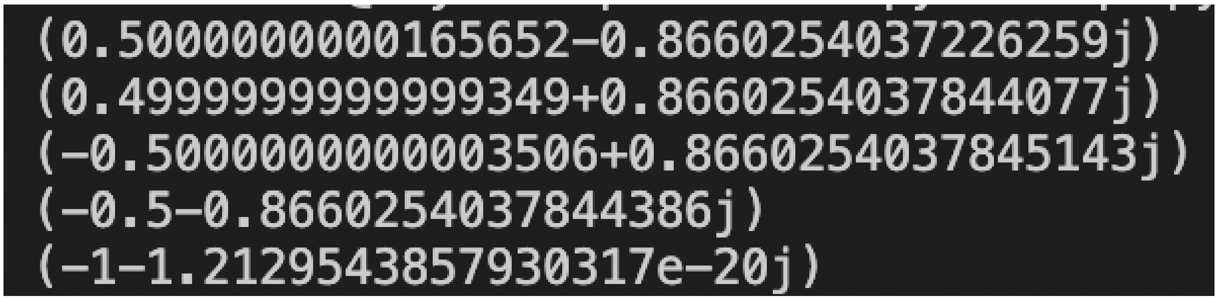
2. quadratic interpolation

$$e_2(x) = \frac{f^{(3)}(\xi)}{3!}(x - x_{i-1})(x - x_i)(x - x_{i+1})$$
$$f^{'''}(x) = -(2\pi)^3 cos(2\pi x) \Rightarrow \max_{0\leq\xi\leq 1}|f^{'''}(\xi)| = (2\pi)^3$$
$$\max_{0\leq x\leq 1}(x - x_{i-1})(x - x_i)(x - x_{i+1}) = \max_{-h\leq y\leq h}|(y+h)y(y-h)| = \frac{2h^3}{3\sqrt{3}}$$
$$\max_{0\leq x\leq 1}|e_2(x)| \leq \frac{1}{6}\cdot(2\pi)^3\cdot\frac{2h^3}{3\sqrt{3}} = \frac{8\pi^3 h^3}{9\sqrt{3}}$$
$$\text{want } \frac{8\pi^3 h^3}{9\sqrt{3}} \leq 5\cdot 10^{-7} \Rightarrow h \leq 0.0031555702$$
$$\Rightarrow N = \frac{1}{h} \approxeq 316.9$$

Therefore, need 318 data points (or 317 intervals).

5.

The implementation is in the q5.py. Run *python q5.py* to get the approximation result.
The approximation result for $x^5 + x^4 + x^3 + x^2 + x + 1$ is:
$(0.5000000000165652 - 0.8660254037226259j)$
$(0.4999999999999349 + 0.8660254037844077j)$
$(-0.5000000000003506 + 0.8660254037845143j)$
$(-0.5 - 0.8660254037844386j)$
$(-1 - 1.2129543857930317e - 20j)$



```
(0.5000000000165652-0.8660254037226259j)
(0.4999999999999349+0.8660254037844077j)
(-0.5000000000003506+0.8660254037845143j)
(-0.5-0.8660254037844386j)
(-1-1.2129543857930317e-20j)
```

7.

Treat x as constant, and rewrite:

$$p(y) = 1 \cdot y^2 + 0 \cdot y + (x^2 - 1)$$
$$q(y) = 0 \cdot y^2 + 1 \cdot y + x^2$$

Then using the method of resultants, define Q =
$\begin{pmatrix} 1 & 0 & x^2-1 & 0 \\ 0 & 1 & 0 & x^2-1 \\ 0 & 1 & x^2 & 0 \\ 0 & 0 & 1 & x^2 \end{pmatrix}$
let $det(Q) = x^4 + x^2 - 1 = 0$, we find the projection points $x = \pm\sqrt{\frac{-1+\sqrt{5}}{2}}$.
By plugging into $p(y) = q(y)$, we see roots $x = \sqrt{\frac{-1+\sqrt{5}}{2}}, y = -\frac{-1+\sqrt{5}}{2}$, and $x = -\sqrt{\frac{-1+\sqrt{5}}{2}}, y = -\frac{-1+\sqrt{5}}{2}$.