16-811 HW2

Dapeng Zhao (Eagle)
AndrewID: dapengz
26 Sep 2019

********** **Q1** **********

a) .
Implemented in *q1.py* .

b)

$$p_5(\frac{1}{3}) = 0.7165316$$

c)

$$\begin{aligned} p_2(0.05) &= 0.99764705 \\ p_4(0.05) &= 0.99013644 \\ p_{40}(0.05) &= 0.96153843 \\ f(0.05) &= 0.7165316 \end{aligned}$$

d)

$$\begin{aligned} E_2 &= 0.574 \text{ at } x = 0.44 \\ E_4 &= 0.385 \text{ at } x = -0.80 \\ E_6 &= 0.489 \text{ at } x = -0.88 \\ E_8 &= 0.732 \text{ at } x = -0.92 \\ E_{10} &= 1.177 \text{ at } x = 0.94 \\ E_{12} &= 1.966 \text{ at } x = 0.95 \\ E_{14} &= 3.380 \text{ at } x = -0.96 \\ E_{16} &= 5.884 \text{ at } x = -0.97 \\ E_{18} &= 10.463 \text{ at } x = -0.97 \\ E_{20} &= 18.378 \text{ at } x = 0.97 \\ E_{40} &= 20258.908 \text{ at } x = 0.99 \end{aligned}$$

Explanation:
The error doesn't converge as the number of interpolation points increases. Having more points to interpolate from is not necessarily a good thing. When interpolation has too many points, the corresponding polynomial function would have too much non-linearity, in which case even though we can force the function to be close to the sample data points, but on the edge of the function range, the error can be comparatively larger.

********** **Q2** **********

- Linear Interpolation:

$$\begin{aligned} &|f''(\xi)| \leq \max_{[-\frac{\pi}{2}, \frac{3\pi}{2}]} |f''(x)| = \max_{[-\frac{\pi}{2}, \frac{3\pi}{2}]} |-cos(x)| = 1 \\ &|(\bar{x} - x_i)(\bar{x} - x_{i+1})| \leq \max_{[0,h]} |(h-y)y| = \frac{h^2}{4} \\ &e_1(\bar{x}) = \frac{|f''(\xi)|}{2!}|(\bar{x} - x_i)(\bar{x} - x_{i+1})| \\ &\qquad \leq \frac{(1)}{2!}(\frac{h^2}{4}) \leq 5 * 10^{-7} \\ &\therefore\ h \approx 0.002 \\ &[\frac{3\pi}{2} - (-\frac{\pi}{2})]/h \approx 3141.6 \end{aligned}$$

Therefore, 3242 intervals and 3243 entries are needed.

- Quadratic Interpolation:

$$\begin{aligned} &|f''(\xi)| \leq \max_{[-\frac{\pi}{2}, \frac{3\pi}{2}]} |f'''(x)| = \max_{[-\frac{\pi}{2}, \frac{3\pi}{2}]} |sin(x)| = 1 \\ &|(\bar{x} - x_{i-1})(\bar{x} - x_i)(\bar{x} - x_{i+1})| \\ &\qquad \leq \max_{[-h,h]} |(h-y)y(h+y)| \\ &\qquad = \frac{2h^2}{3\sqrt{3}} \\ &e_1(\bar{x}) = \frac{|f'''(\xi)|}{3!}|(\bar{x} - x_{i-1})(\bar{x} - x_i)(\bar{x} - x_{i+1})| \\ &\qquad \leq \frac{(1)}{3!}(\frac{2h^2}{3\sqrt{3}}) \\ &\qquad \leq 5 * 10^{-7} \\ &\therefore\ h \approx 0.019827 \\ &[\frac{3\pi}{2} - (-\frac{\pi}{2})]/h \approx 316.9 \end{aligned}$$

Therefore, 317 intervals and 318 entries are needed.

********** **Q3** **********

The solutions are 4.49340946 and 7.72525184 .
Implemented in *q3.py*.
The ending condition is $|f(x)| < 10^{-6}$.

********** Q4 **********

Denote:

$$h(x) = \frac{f}{f'}$$

$$\begin{aligned}
\lim_{x\to\xi} h(x) &= \lim_{x\to\xi} \frac{f}{f'} = \lim_{x\to\xi} \frac{f'}{f''} = 0 \\
\lim_{x\to\xi} h'(x) &= \lim_{x\to\xi} (1 - \frac{ff''}{f'f'}) \\
&= 1 - \lim_{x\to\xi} (\frac{f'f'' + ff'''}{2f'f''}) \\
&= 1 - \lim_{x\to\xi} (\frac{f'f''' + f''f'' + ff'''' + f'f'''}{2f'f''' + 2f''f''}) \\
&= 1 - \lim_{x\to\xi} (\frac{f''f''}{2f''f''}) \\
&= 1 - \frac{1}{2} \\
&= \frac{1}{2} \\
\lim_{x\to\xi} h''(x) &= \lim_{x\to\xi} (-\frac{(f'f'' + ff''')f'f' - ff''(2f'f'')}{f'f'f'f'}) \\
&= \lim_{x\to\xi} (-\frac{f'f'f'' + ff'f''' - 2ff''f''}{f'f'f'}) \\
&= \lim_{x\to\xi} (-\frac{2f'f'f''' + ff'f'''' - 3ff''f'''}{3f'f'f''}) \\
&= \lim_{x\to\xi} (-[3f'f'f'''' + f'f''f''' + ff'f'''' \\
&\qquad - 3ff'''f''' - 2ff''f''''] \\
&\qquad /[3(f'f'f''' + 2f'f''f'')]) \\
&= \lim_{x\to\xi} (-\frac{f''f''f'''}{3(2f''f''f'')}) \\
&= \lim_{x\to\xi} (-\frac{f'''}{6f''})
\end{aligned}$$

When $x_{n+1} = x_n - h(x_n)$:

$$\begin{aligned}
(\xi + \varepsilon_{n+1}) &= (\xi + \varepsilon_n) - h(x_n) \\
\varepsilon_{n+1} &= \varepsilon_n - h(x_n) \\
\lim_{\varepsilon_n\to 0} \varepsilon_{n+1} &= \lim_{\varepsilon_n\to 0} (\varepsilon_n - [h(\xi) + h'(\xi)\varepsilon_n]) \\
&= \lim_{\varepsilon_n\to 0} (\varepsilon_n - [0 + \frac{1}{2}\varepsilon_n]) \\
&= \lim_{\varepsilon_n\to 0} \frac{1}{2}\varepsilon_n \\
\lim_{\varepsilon_n\to 0} \frac{\varepsilon_{n+1}}{\varepsilon_n} &= \frac{1}{2}
\end{aligned}$$

The order of convergence is 1, indicating **linear convergence**.

When $x_{n+1} = x_n - 2h(x_n)$:

$$\begin{aligned}
\varepsilon_{n+1} &= \varepsilon_n - 2h(x_n) \\
\lim_{\varepsilon_n\to 0} \varepsilon_{n+1} &= \lim_{\varepsilon_n\to 0} (\varepsilon_n - 2[h(\xi) + h'(\xi)\varepsilon_n + \frac{h''(\xi)}{2!}\varepsilon_n^2]) \\
&= \lim_{\varepsilon_n\to 0} (\varepsilon_n - 2[0 + \frac{1}{2}\varepsilon_n + \frac{1}{2!}(-\frac{f'''}{6f''})\varepsilon_n^2]) \\
&= \lim_{\varepsilon_n\to 0} -\frac{f'''}{6f''}\varepsilon_n^2 \\
\lim_{\varepsilon_n\to 0} \frac{\varepsilon_{n+1}}{\varepsilon_n^2} &= -\frac{f'''}{6f''}
\end{aligned}$$

The order of convergence is 2, indicating **quadratic convergence**.

.

.

********** Q5 **********

a) .

Implemented in *q5.py* .

Firstly I found one root with Muller's method, and then I define a new function $g(x)$ as $\frac{f(x)}{(x-r_0)}$, where r_0 is the first found root. Then I perform Muller's method on $g(x)$ to find the next root, and the third root is found similarly.

b) .

The roots are 3, 1-2i, and 1+2i .

The later two roots are a conjugate pair.

.

.

********** Q6 **********

a)

$$R = \begin{pmatrix} 1 & -4 & 6 & -4 & 0 \\ 0 & 1 & -4 & 6 & -4 \\ 1 & 2 & -8 & 0 & 0 \\ 0 & 1 & 2 & -8 & 0 \\ 0 & 0 & 1 & 2 & -8 \end{pmatrix}$$

$det(R) = 0$, therefore, common root exsists between $p(x)$ and $q(x)$.

b)

$$\vec{x} = \begin{pmatrix} x^4 \\ x^3 \\ x^2 \\ x \\ 1 \end{pmatrix}$$

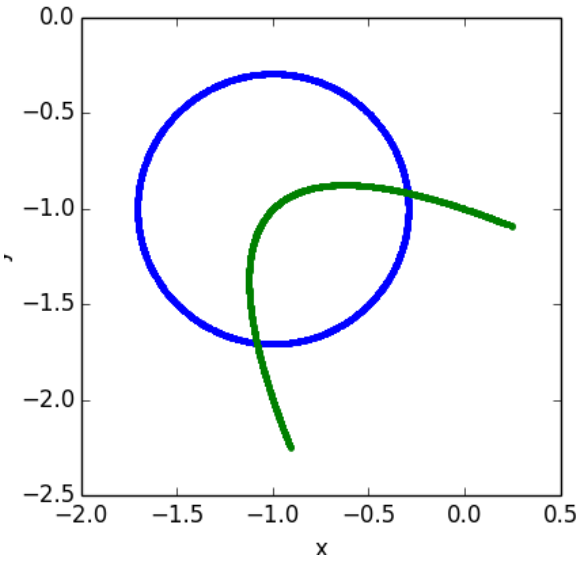
$$R\vec{x} = \vec{0}$$

$$\frac{x_1}{x_2} = \frac{x^4}{x^3} = (-1)^{1+2}\frac{det(A_1)}{det(A_2)} = (-1)\frac{832}{-416} = 2$$

The common root is $x = 2$

********** Q7 **********

a) .



b)

$$Q = \begin{pmatrix} 2 & 4 & 2x^2+4x+3 & 0 \\ 0 & 2 & 4 & 2x^2+4x+3 \\ 1 & 2x+5 & x^2+3x+4 & 0 \\ 0 & 1 & 2x+5 & x^2+3x+4 \end{pmatrix}$$

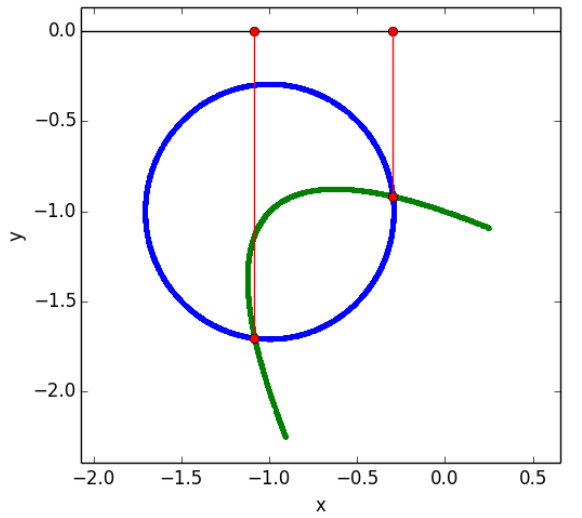
$$det(Q) = 16x^4 + 80x^3 + 144x^2 + 100x + 10 = 0$$

$$x = [-0.2979073, -1.0840587]$$

For each root for x slove y for both equations p and q:

$$y = [-0.9159413, -1.7020927]$$

c) .



********** Q8 **********

a) .

Implemented as *getWeights(X,p)* in *q8.py*.
The method is to solve:

$$\begin{pmatrix} x^{(i)} & x^{(j)} & x^{(k)} \\ y^{(i)} & y^{(j)} & y^{(k)} \\ 1 & 1 & 1 \end{pmatrix} \vec{v} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

If elements of $\vec{v}$ are all positive, the point (x,y) is inside of the triangle.

b) .

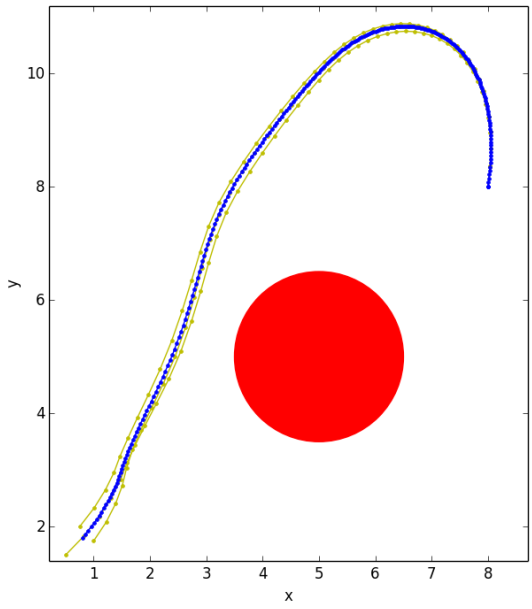
Implemented in *q8.py*.

c) .

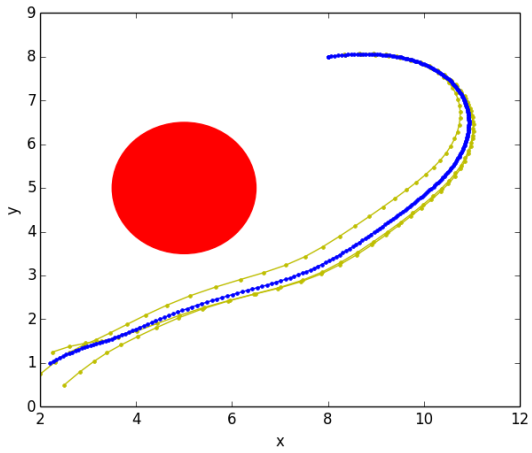
All the given paths are divided into two groups based on whether the path go to left or right side to avoid the ring of fire. From each group, 3 paths are selected for having starting locations that (1) are the closest to our given starting location, (2) form a triangle which our starting location falls within. The weights are generated by function *getWeights(X,p)*, which is the Barycentric coordinates of our given starting location in the system of 3 picked points. The time scale t range $(0, 49)$ because this range is natrually the range of the indices of path coordinates list. Linear interpolation was chosen because of its simplicity – if the simple approach works good enough, there is no need for trying more complex approaches.

d) .

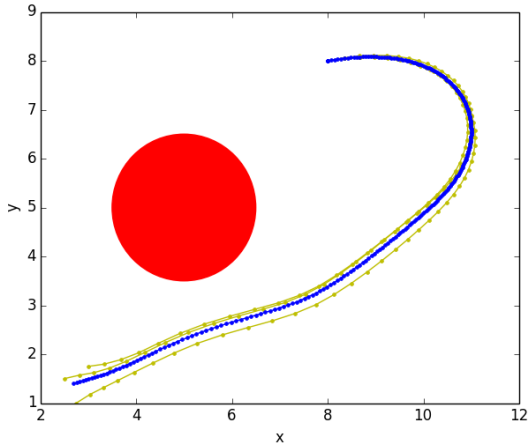
The yellow dot-lines are picked 3 paths, and the blue dot-line are generated by interpolation.
Starting from (0.8, 1.8):



Starting from (2.2, 1.0):



Starting from (2.7, 1.4):



e) .

Currently, because of this one obstacle, ring of fire, the given sample paths naturally have to either go to left or right to avoid obstable, thus there formed two groups of paths. The algrithm has to make sure it is generating new path from three sample paths which belong to the same group.

If more obstacles exist, the paths would have even more kinds of general shapes, meaning they would have to be divided into more groups based on how they each avoid obstacles. The sample paths grouping part of my algorithm would need to be updated so that I can make sure the generated new path is not a weighted average of paths that have different shapes, that the selected three sample paths avoid each obstacle to the same direction.

.

.