16-811 Math Fundamentals for Robotics
Assignment 2

Sajal Maheshwari
sajalm@andrew.cmu.edu

September 11, 2019

1 Collaborations

1.1 Question 4

For Question 4, I used an online resource to understand the problem and verify my solution ((a) : if neglecting the double derivative is fine or not and (b)for proving that the solution after changing the update rule, is it sufficient to just show that the linear term has become zero.)

The URL used is : `https://homepage.divms.uiowa.edu/~atkinson/ftp/ENA_Materials/Overheads/sec_3-5.pdf?fbclid=IwAR2Yr75U9mnqExhYIaerSj8V1VbrtwBu73Y8k3OAgUJjGNw`

2 Instructions for Running code

All codes are written in MATLAB.

2.1 Question 1

For question 1, the code is broken into multiple sub-sections separated by comments, where each section gives the output of one of the parts of the questions. The function 'NewtonInterp' is used to calculate the interpolation and is utilized in each of the questions as and when required.

2.2 Question 3

For question 3, at the start of the code, I declare the various hyperparameters used in the solution. The startBracket variable is set to 7, as we need to find roots around that. The endBracket variable is set to the regions in which we search for the start point for the Newton Method.

The precision variable is the precision the code is willing to accomodate while root finding. The tolerance is the value of the function which we deem close enough for the Root-finding algorithm to start.

We first find the value within our search interval at which the function output is less than the tolerance. This is chosen as the starting value. Then the root-finding algorithm is applied till the function output is within precision range of the root.

The expression for the function and its differential are also specified. The evaluator can change these variables for verification as per his/her convenience.

2.3 Question 5

For question 5, at the start of the code, the function is defined. Then the inital starting point roots are selected in the next three variables, before we call the algorithm. After a root is found we deflate the function and iterate through the same processa again.

2.4 Question 7

In question 7, the two equations are declared using the symbolic package of MATLAB. Thereafter, we declate the Sylvester matrix using the coefficients of the two expressions (assuming x as a constant). The standard resultant calculation is then followed, with the determinant of Sylvester matrix set to zero, which outputs the values of x.

These values are then plugged in both the functions individually to find the common values of y. This gives us the common roots, which are displayed on the command line.

2.5 Question 8

The function 'isPointInTriangle', as the name suggests returns a boolean variable, which is true if a given point is inside the trianle formed by three other points. These 4 points are passed as arguments to the function in the following order - The point whose insideness/outsideness w.r.t the triangle to be determined, vertex1 of triangle, vertex2 of triangle, vertex3 of triangle.

This function is utilized later where we search for the paths whose starting points form a triangle such that the unicycle start point lies within them. We select such triangles whose all paths are on the same side of the line $y = x$. And then use this to get the desired path. The interpolation method used to get the continuous polynomial is spline fitting.

Q1. (b)

Command Window

New to MATLAB? See resources for Getting Started.

```
n =

     2

    0.0500    0.9976


realOut =

    0.9615

>>
```

Q1. C

```
Command Window
New to MATLAB? See resources for Getting Started.

n =

     2

    0.0500    0.9976


realOut =

    0.9615

fx >>
```

```
Command Window
New to MATLAB? See resources for Getting Started.

n =

     4

    0.0500    0.9901


realOut =

    0.9615

fx >>
```

```
Command Window
New to MATLAB? See resources for Getting Started.

n =

    40

    0.0500    0.9615


realOut =

    0.9615

fx >>
```

Q1. d

Value of n : 2

```
    0.5736   -0.4343
```

Value of n : 4

```
    0.3853   -0.7980
```

Value of n : 6

```
    0.4883    0.8788
```

Value of n : 8

```
    0.7318    0.9192
```

Value of n : 10

```
    1.1767   -0.9394
```

Value of n : 12

```
    1.9423   -0.9596
```

Value of n : 14

```
    3.3790    0.9596
```

Value of n : 16

```
    5.7551    0.9596
```

Value of n : 18

```
    9.8790    0.9798
```

Value of n : 20

```
   18.3280   -0.9798
```

Value of n : 40

```
   1.0e+03 *

    6.3337    0.0010
```

Q3.

Command Window

New to MATLAB? See resources for Getting Started.

```
First root :
    7.7253

Second root :
    4.4934

>>
```

Q5.

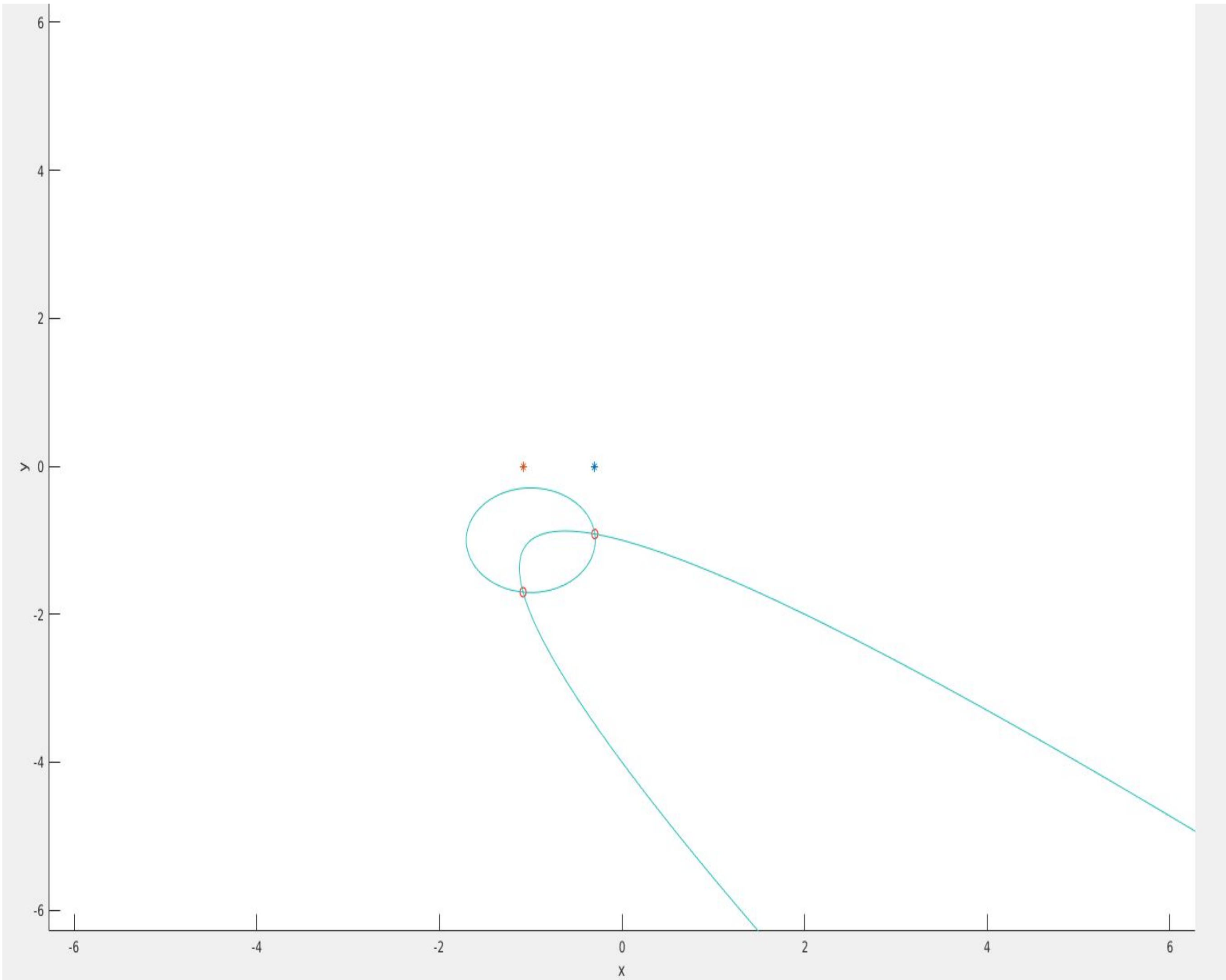

Command Window
New to MATLAB? See resources for Getting Started.
First root :
1.0000 - 2.0000i
Second root :
3.0000 + 0.0000i
Third root :
1.0000 + 2.0000i
>>

Q7.

Q7 Steps :

```
Command Window
New to MATLAB? See resources for Getting Started.

Sylvester matrix :

SylvesterMatX =

[ 2*x^2 + 4*x + 3,               4,       2, 0]
[               0, 2*x^2 + 4*x + 3,       4, 2]
[   x^2 + 3*x + 4,         2*x + 5,       1, 0]
[               0,   x^2 + 3*x + 4, 2*x + 5, 1]

Sylvester matrix determinant:

detSylvester =

16*x^4 + 80*x^3 + 144*x^2 + 100*x + 19

Root of determinant of Sylvester Matrix :

valuesX =

   -0.2979
   -1.0841
```

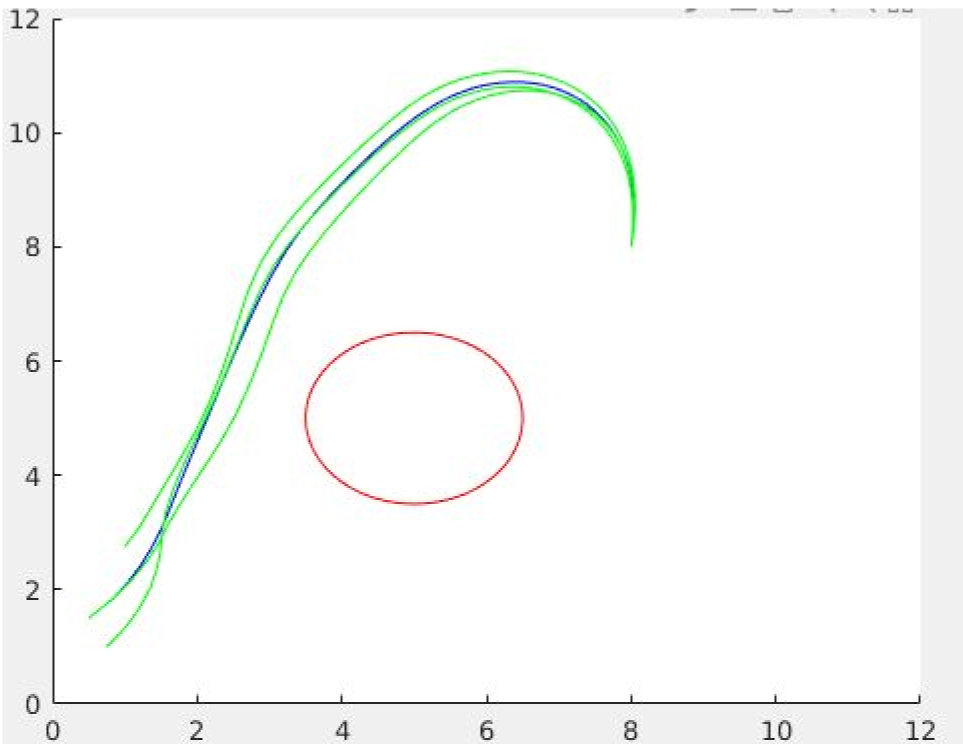
Q7. Output graph :



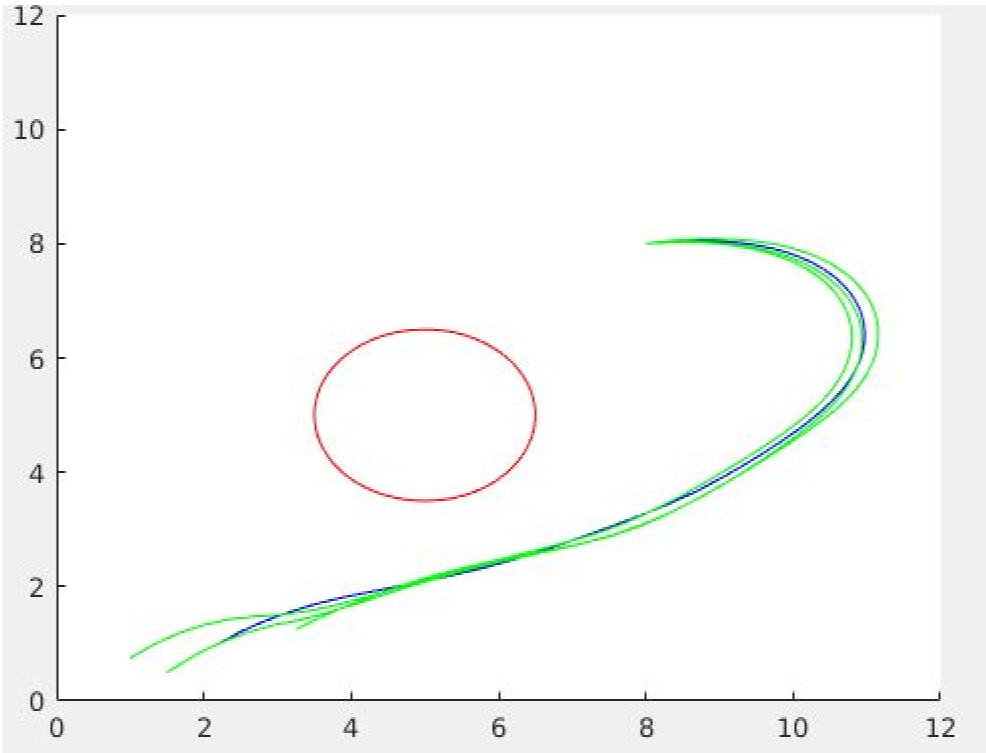
Q8 :

The red circle represents the fire pit.
The green lines represent the pre-defined paths used for the new path tracing.
The blue line represents the traced path.

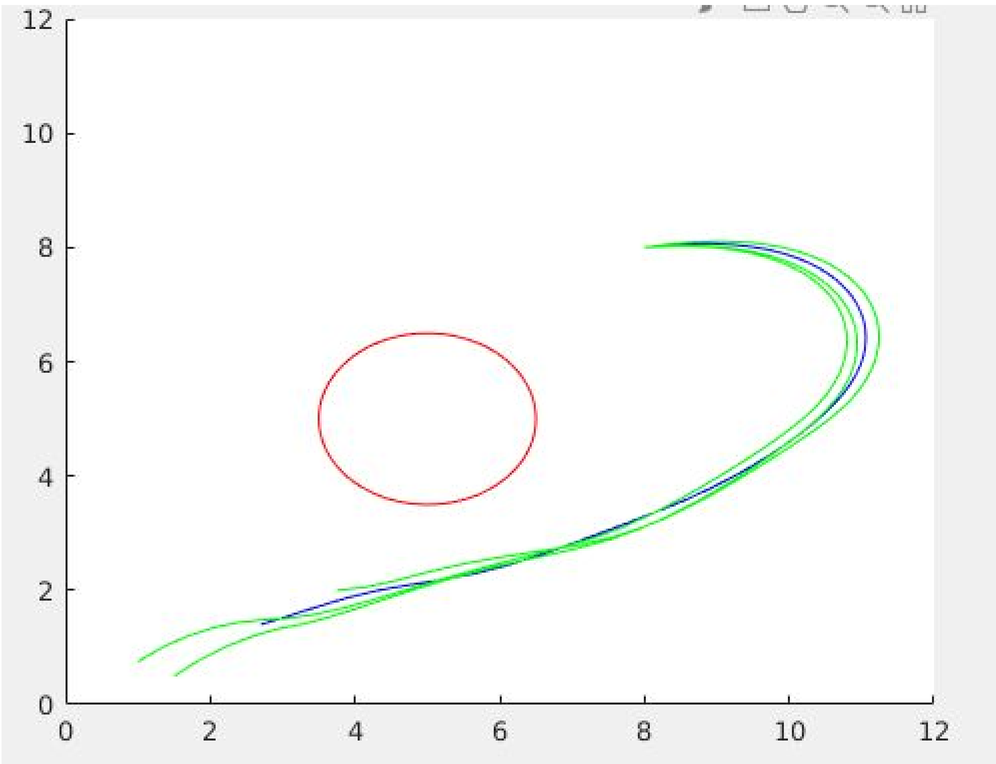
Point : [0.8; 1.8]



Point : [2.2; 1.0]



Point : [2.7, 1.4]



Q.1. (d) We observe that the error increases as we increase the number of points. This is because we try to fit a polynomial, while the true func^n is not a polynomial. $\therefore$ as the degree of the polynomial that is being fitted increases we start getting higher error, as the polynomial is highly unstable and non-smooth.

Q.2 → $f(x) = \cos(x)$. $x \in [-\pi/2, 3\pi/2]$

We need to find the table spacing for 6 decimal digit accuracy.

To find this, we make use of the theorem [explained in class]:

If $f: [a,b] \to \mathbb{R}$ is $(n+1)$ times differentiable. If $p_n(x)$ interpolates $f(x)$ at $x_0, \dots, x_n$, then for every $\bar{x} \in [a,b]$, we have $\xi \in (a,b)$ such that

$$e_n(\bar{x}) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{i=0}^{n} (\bar{x} - x_i)$$

For <u>linear interpolation</u>

$$e_1(x) = \frac{f^{(1+1)}(\xi)}{(1+1)!} \prod_{i=0}^{1} (\bar{x} - x_i)$$

$$= \frac{f''(\xi)}{2!} (\bar{x} - x_0)(\bar{x} - x_1)$$

Now, $\because$ we do not know ξ, we can not find exact error. So, we try to find the lower bound

$$e_1(x) \leq \underbrace{\max\left(\frac{f''(\xi)}{2!}\right)}_{1} \cdot \underbrace{\max\left((\bar{x}-x_0)(\bar{x}-x_1)\right)}_{2}$$

For linear interpolation, we ~~keep the~~ first find the max bound on 1

$$\max\left(\frac{f''(\xi)}{2!}\right)$$

$$= \frac{1}{2}\max\left(f''(\xi)\right)$$

$$f(x) = \cos(x)$$

$$f'(x) = -\sin(x)$$

$$f''(x) = -\cos(x)$$

$$\max_{-\pi/2 \leq x \leq 3\pi/2} (f''(x)) = 1$$

$\Rightarrow$ The max. value of 1^{st} expression is $\frac{1}{2}$.

For the 2^{nd} expression

$$\max_{x \in [0,h]} |(\bar{x}-h)(\bar{x})|$$

[For an interval, $(\bar{x}-h)$ to $(\bar{x})$, this is the polynomial which interpolates]

$$= |(h-h/2)(h/2)|$$

$$= h^2/4$$

Now, to get the bound on accuracy (10^{-6}) the expression to solve becomes:

$$\frac{1}{2} \times \frac{h^2}{4} \leq 5 \times 10^{-7}$$

$$\frac{h^2}{\cancel{8}} \leq 40 \times 10^{-7}$$

$$h^2 \leq 4 \times 10^{-6}$$

$$h \leq 2 \times 10^{-3}$$

The spacing we want is of (2×10^{-3}).

$\Rightarrow$ The ~~intervals~~ points in beth the end points are of size $\left\lceil \frac{2\pi}{2\times10^{-3}} \right\rceil$

$\Rightarrow$ The points in beth the end points $= \lceil 1000\pi \rceil$

$= 3142$. Ans.

For Quadratic interpolation:

The first expression changes to

$$\frac{f'''(\xi)}{3!}$$

We know, $f''(x) = -\cos(x)$

$\Rightarrow f'''(x) = \sin(x)$

$$\max_{-\pi/2,\, 3\pi/2} (f'''(x)) = 1$$

$$\Rightarrow \max\left(\frac{f'''(\xi)}{3!}\right) = \frac{1}{3!} = \frac{1}{6}.$$

The second expression is:

$$\prod_{i=0}^{2} (\bar{x} - x_i)$$

$$= (\bar{x}-x_0)(\bar{x}-x_1)(\bar{x}-x_2)$$

Which in the max bound finding case is:

$$\max_{-h\le y\le h} |(y-h)\,y\,(y+h)|$$

$$= \frac{2}{3}\times\frac{1}{\sqrt{3}}h^3$$

The expression to solve becomes

$$\frac{1}{6}\times\frac{2}{3}\times\frac{1}{\sqrt{3}}h^3 \le 5\times10^{-7}$$

$$h^3 \le 45\sqrt{3}\times10^{-7} \Rightarrow h \approx [illegible].$$

$h = \lceil (45\sqrt{3} \times 10^{-7})^{1/3} \rceil$

$\Rightarrow$ the no. of intervals is

$$\frac{2\pi}{\lceil h \rceil} = \frac{2\pi}{(45\sqrt{3} \times 10^{-7})^{1/3}}$$

$$= 318. \quad \underline{\underline{\text{Ans}}}.$$

Q.4. $$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \quad \text{—} \quad 2$$

Let us consider $h(x) = \dfrac{f(x_n)}{f'(x_n)}$

We expand $h(x)$ using Taylor series at $x = \xi_1$

$$h(\xi_1 + \epsilon) = h(\xi_1) + \epsilon h'(\xi_1) + \frac{\epsilon^2}{2} h''(\xi_1) + \ldots$$

We first compute $h'(\xi_1)$

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

$$\frac{d}{dx}(h(x)) = \frac{d}{dx}\left(\frac{f(x)}{f'(x)}\right)$$

$$= \frac{f'(x).\ f'(x) - f(x).\ f''(x)}{(f'(x))^2}$$

$$= \frac{(f'(x))^2}{(f'(x))^2} - \frac{f(x) f''(x)}{(f'(x))^2}$$

$$= 1 - \underbrace{\frac{f(x) f''(x)}{(f'(x))^2}}_{0/0 \text{ form at } x = \xi_1}$$

$\therefore$ Apply L' Hospital Rule.

L' Hospital applied again.

$$= 1 - \lim_{x \to \xi_1} \left[\frac{f(x) f'''(x) + f''(x) f'(x)}{2 f'(x) f''(x)}\right]$$

$$= 1 - \frac{1}{2} \lim_{x \to \xi_1} \left[\frac{\cancelto{0}{f(x)} f''''(x) + \cancelto{0}{f'(x)} f'''(x) + \cancelto{0}{f'''(x) f'(x)} + (f''(x))^2}{(f''(x))^2 + \cancelto{0}{f'(x) f'''(x)}}\right]$$

$$h'(x)\Big|_{x=\xi_1} = 1/2$$

$\therefore$ $h(\xi_1 + \epsilon) = h(\xi_1) + \frac{\epsilon}{2}$ + insignificant terms that can be ignored.

~~In Newton's method, $x_n - \xi_1 = \epsilon_n$~~

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

Now, $x_n = \xi_1 - \epsilon_n$

$$\Rightarrow h(x_n) = h(\xi_1 - \epsilon_n)$$
$$= h(\xi_1) + \epsilon_n/2$$
$$= \epsilon_n/2$$

$$\Rightarrow \epsilon_{n+1} \approx |\epsilon_n/2|$$

$\therefore$ we ~~can~~ show that the Newton's method is converging linearly.

For, the expression (1) if we change the func^n to:

$$x_{n+1} = x_n - 2\frac{f(x_n)}{f'(x_n)}$$

$\frac{d}{dx}\left(2\frac{f(x_n)}{f'(x_n)}\right)\Big|_{x_n=\xi_1}$ becomes:

$$= 1 - \frac{2f(x)f''(x)}{[f'(x)]^2} = 1 - \frac{2\times}{2}\left[\frac{f''(x)}{f''(x)}\right]^2 = 0$$

(SEE LAST EXPRESSION ON PREVIOUS PAGE)

Thus, $h'(\varepsilon_i)$ becomes 0.

This makes the func^n back to quadratic convergent.

Q.6 - $p(x) = x^3 - 4x^2 + 6x - 4$
$q(x) = x^2 + 2x - 8$

(a) We construct the matrix Q.

$$Q = \begin{bmatrix} 1 & -4 & 6 & -4 & 0 \\ 0 & 1 & -4 & 6 & -4 \\ 1 & 2 & -8 & 0 & 0 \\ 0 & 1 & 2 & -8 & 0 \\ 0 & 0 & 1 & 2 & -8 \end{bmatrix}$$

The $\det(Q) = 0$
$\because \det(Q) = 0$, we will have a common root. Ans.

(b) We use the ratio method to find the common root. In this method, we look at the eqn.

$$\begin{bmatrix} 1 & -4 & 6 & -4 & 0 \\ 0 & 1 & -4 & 6 & -4 \\ 1 & 2 & -8 & 0 & 0 \\ 0 & 1 & 2 & -8 & 0 \\ 0 & 0 & 1 & 2 & -8 \end{bmatrix} \begin{bmatrix} x^4 \\ x^3 \\ x^2 \\ x \\ 1 \end{bmatrix} = [0]$$

To find the ratio $\left(\frac{x^2}{x} = x\right)$, we remove the last row to get matrix

$$M = \begin{bmatrix} 1 & -4 & 6 & -4 & 0 \\ 0 & 1 & -4 & 6 & -4 \\ 1 & 2 & -8 & 0 & 0 \\ 0 & 1 & 2 & -8 & 0 \end{bmatrix}$$

Remove this column to get $\det(x)$

$\det(A)$ represents the column by removing the corresponding column.

Remove this column to get $\det(x^2)$

$\det(x) = 52$
$\det(x^2) = -104$

$$\frac{x^2}{x} = (-1)^{2+3} \frac{\det(x^2)}{\det(x)}$$

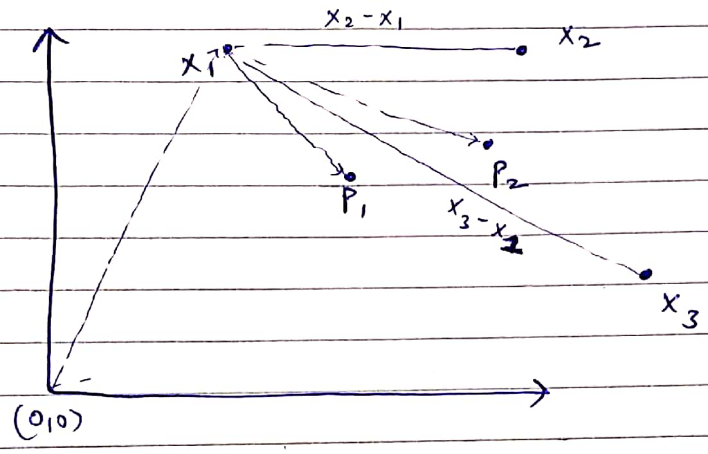
$$= (+1) \times \frac{(+\cancel{104})}{\cancel{52}} = 2$$

$x = 2$

Thus, the common root is 2. Ans.

Q.8 (a) We need to write a set of equations such that, given 3 2D points, we can judge if a new point lies in the triangle formed by the previous 3 points.

We look at the graph below:



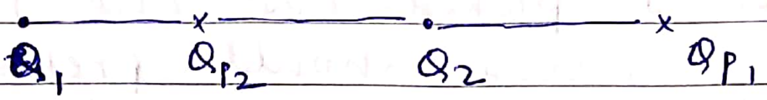
$\therefore$ we can represent any 2D point as a span of 2 independent vectors in 2D,

we choose these vectors to be $(\bar{x}_3 - \bar{x}_1)$ and $(\bar{x}_2 - \bar{x}_1)$.

We Specifically choose the difference in sides, Because, $(\bar{x}_2 - \bar{x}_1)$ AND $(\bar{x}_3 - \bar{x}_1)$ are bound to be Independent of each other if a triangle exists.

Now, we are able to represent any point in 2D space as a combination of these 2 vectors.

We look at another graph now.



We can represent both Q_{P_1} and Q_{P_2} in terms of Q_1 and Q_2, using section Formula. However, the ratios will both be less than 1 ~~if~~ for Q_{P_2}, as it lies between the 2 points Q_1 and Q_2. This is not the case with Q_{P_1}.

The same logic can be generalized for 2D.

$\Rightarrow$ If a vector $\bar{v} = \alpha(\bar{x}_2 - \bar{x}_1) + \beta(\bar{x}_3 - \bar{x}_1)$

$\boxed{\text{is such that } \alpha \in [0,1] \text{ and } \beta \in [0,1] \text{ and } (\alpha+\beta) \in [0,1]}$

then $\bar{v}$ lies between $(\bar{x}_2 - \bar{x}_1)$ and $(\bar{x}_3 - \bar{x}_1)$.

$\therefore$ our linear equation system becomes

$$\boxed{\begin{array}{c} \begin{bmatrix} (x_2^1 - x_1^1) & (x_3^1 - x_1^1) \\ (x_2^2 - x_1^2) & (x_3^2 - x_1^2) \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} p^1 - x_1^1 \\ p^2 - x_2^1 \end{bmatrix} \\ \text{where } \alpha \in [0,1] \text{ and } \beta \in [0,1]. \quad \underline{\text{Ans}}. \\ \text{and } \alpha+\beta \in [0,1] \end{array}}$$

(b) CODE

(c) The triple path I picked had one pre-condition.
- The paths to choose should preferrably be lying ~~via~~ on the same side of $(y=x)$ line.

This path will automatically ensure that we DO NOT go into the fire-pit because:

a) The path will start from a region that lies in the triangle formed by three path on the same side, hence not in fire.

b) For every next time instant, the resultant point will be a weighted sum of three paths definitely not closing the fire pit on the same side, with weights < 1. Thus, the point will not be crossing the fire pit, as it remains in the vicinity of pre-defined paths.

The weights for the proposed algorithm, automatically come out to be

$(1-A-B)$, (A) and (B) where

$$\begin{bmatrix} (x_2^1 - x_1^1) & (x_3^1 - x_1^1) \\ (x_2^2 - x_1^2) & (x_3^2 - x_1^2) \end{bmatrix} \begin{bmatrix} A \\ B \end{bmatrix} = \begin{bmatrix} P_1^1 - x_1^1 \\ P_1^2 - x_1^2 \end{bmatrix}$$

is the equation we solve.

The proposed method uses spline interpolation as:

i) We can represent the function as a low-degree polynomial (Having a 49 degree polynomial using 50 points will be highly noisy and overfit.

ii) The spline interpolation is piece-wise (which is the nature of the problem in this case, the next step of the rider is not dependent on his coordinates long time ago.

iii) The spline interpolation is differentiable, unlike piece-wise linear interpolation.

CODE.

If more obstacles will be included on the line (x, y), the change in algorithm will depend on the space occupied by the obstacle.

If the obstacle is at another location, we will have to introduce more constraints on the start point, so that ~~it is far~~ the paths selected are quite far away from all the obstacles