16-811 Assignment 2

Emma Benjaminson

26 September 2019

1 Problem 1

NOTE: I used the following references to help me solve this problem:

(1) Kaw, A. and Keteltas, M. "Newton's Divided Difference Interpolation." 23 Dec 2009. `http://mathforcollege.com/nm/mws/gen/05inp/mws_gen_inp_txt_ndd.pdf`

(2) Erdmann, Michael. "Polynomial Approximations - Interpolation." Course notes from Fall 2019.

1.1 Part (a)

Please see code located in code/q1.m for the implementation of my divided differences function.

1.2 Part (b)

Please see code located in code/q1.m for the solution to this problem. I used my divided differences function to calculate an interpolated value of 0.7142 for $x = \frac{1}{3}$, which matches the correct answer, 0.7165, to two decimal places.

1.3 Part (c)

Please see code located in code/q1.m for the full solution to this problem. I have plotted the interpolated and true values against the number of points used to interpolate in Figure 1. The actual value of $f(0.05) = 0.9615$. I found that increasing the number of sample points in this instance worked to improve the accuracy of my interpolated values. I also found that my interpolated values oscillated above and below the true value, I did not approach the true value from below or above uniformly.

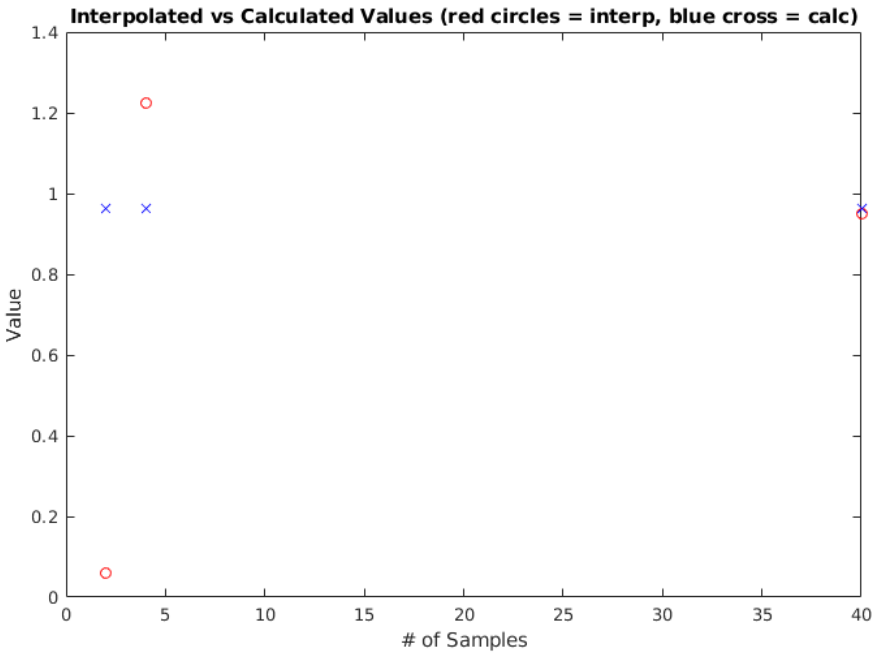

Figure 1: Plot of interpreted and actual values for sample sizes of 2, 4 and 40.

1.4 Part (d)

Please see the code located in code/q1.m for the full solution to this problem. I have plotted the error estimates for the given function against the number of sample points in Figure 2 below. The plot shows that the maximum possible interpolation error is the worst for 6 sample points, and decreases exponentially after that number of samples (the error increases linearly up to 6 samples).
I think it makes sense that the error estimates would decrease exponentially and asymptotically approach some minimum as the number of samples increased. This makes sense to me because if the interval over which we are calculating error is bounded, then we should quickly decrease our error to appear at smaller and smaller decimal places. But I find it odd that we are not approaching a null difference between the function and the estimate - instead, we seem to be approaching some offset value. I don't know exactly why that is happening - maybe it indicates that this interpolation method has some built-in error or shift away from the true value that we can never overcome.
I am not sure why we see the maximum possible error peak at 6 samples. I wonder if the exact place where the error peaks is function-dependent. I could perhaps understand that the error has some peak for a low (but not minimal) number of samples because a minimal number of samples would simply

interpolate the midpoint of the interval, which might not be that far off from the true value. But a small number of samples (say less than 10, but greater than 2) might provide enough points to interpolate to a value that is actually farther away from the correct value than an interpolated estimate with fewer points would be.

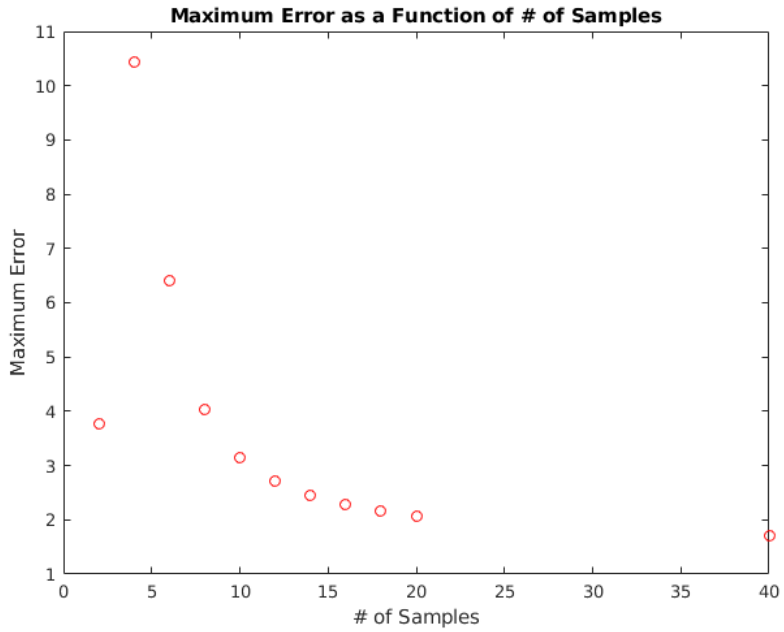

Figure 2: Plot of estimated error for divided differences interpolation function against number of sample points used.

2 Problem 2

NOTE: I used the following references to help solve this problem:

(1) Erdmann, Michael. "Polynomial Approximations - Interpolation." Course notes from Fall 2019.

(2) Qiu, Jingmei. "Polynomial Interpolation - Error Analysis." University of Houston course notes. `https://www.math.uh.edu/~jingqiu/math4364/interp_error.pdf`

In this problem we have:

$$f(x) = cos(x), x \in \big[-\frac{\pi}{2}, \frac{3\pi}{2} \big] \tag{1}$$

The general form expression for the error in our interpolation is:

$$e_n(\bar{x}) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\prod_{i=0}^{n}(\bar{x} - x_i) \tag{2}$$

In this problem, x is a collection of uniformly spaced points over a range of 2π. If I have n points, then the spacing between adjacent points is $h = \frac{2\pi}{n}$. This allows me to say that every point can be written as:

$$x_i = -\frac{\pi}{2} + ih = -\frac{\pi}{2} + i(\frac{2\pi}{n}) \tag{3}$$

My objective in this problem is to find the value of h (the spacing between points) required to achieve 6 decimal digit accuracy. First let's solve this problem for the linear interpolation case (with a second order polynomial) and then expand to the quadratic interpolation case (with a third order polynomial).

2.1 Linear Interpolation Solution

Let's rewrite Eq 2 in terms of a linear interpolation. First we can say that:

$$\frac{f''(\xi)}{2!} = -\frac{cos(\xi)}{2} \tag{4}$$

I also know that my product contains 2 terms as follows:

$$(\bar{x} - x_0)(\bar{x} - x_1) \tag{5}$$

And given my expression for x_i in Eq 3, I can rewrite the entire equation as:

$$e_1(\bar{x}) = -\frac{cos(\xi)}{2}(\bar{x} - (-\frac{\pi}{2}))(\bar{x} - (-\frac{\pi}{2} + h)) \tag{6}$$

Now since I cannot find ξ exactly I need to bound the error by finding what it could be in the worst case. The worst case assumes that each component of the equation is maximized. I am going to find the maximum of the cosine component first as:

$$\max_{-\frac{\pi}{2} \leq x \leq \frac{3\pi}{2}} \left| -\frac{cos(x)}{2} \right| = \frac{1}{2} \tag{7}$$

Now I need to maximize the product:

$$\max_{\bar{x} \in [x_0, x_1]} |(\bar{x} - x_0)(\bar{x} - x_1)| \leq \max_{y \in [-h, h]} |(y + h)(y - h)| \tag{8}$$

$$= \max_{y \in [-h, h]} |y^2 - h^2| \tag{9}$$

Looking at the derivative to find a maximum, we can see that:

$$2y = 0, \forall h \tag{10}$$

The maximum value for this expression is not dependent on h, it occurs when $y = \bar{x} = 0$. In this case, we can say that the worst case error is:

$$e_1(\bar{x}) = \frac{1}{2}(0 + \frac{\pi}{2})(0 + \frac{\pi}{2} - h) = \frac{1}{2}((\frac{\pi}{2})^2 + \frac{\pi}{2}h) \tag{11}$$

If our initial requirement is that the error must be smaller than 6 decimal places, I can write that as:

$$\frac{1}{2}((\frac{\pi}{2})^2 + \frac{\pi}{2}h) \leq 5 \times 10^{-7} \tag{12}$$

This seems to indicate that $h \approx \frac{\pi}{2} \approx 1.5$ sample points. I am not sure if that is correct or not.

2.2 Quadratic Interpolation Solution

I can follow the same logic as presented above:

$$\frac{f'''(\xi)}{3!} = \frac{sin(\xi)}{6} \tag{13}$$

And the product contains 3 terms:

$$(\bar{x} - x_0)(\bar{x} - x_1)(\bar{x} - x_2) \tag{14}$$

Putting it all together I get:

$$e_2(\bar{x}) = \frac{sin(\xi)}{6}(\bar{x} - x_0)(\bar{x} - x_1)(\bar{x} - x_2) \tag{15}$$

Let's maximize each element:

$$\max_{-\frac{\pi}{2} \leq x \leq \frac{3\pi}{2}} \left| \frac{sin(\xi)}{6} \right| = \frac{1}{6} \tag{16}$$

$$\max_{\bar{x} \in [x_0, x_2]} |(\bar{x} - x_0)(\bar{x} - x_1)(\bar{x} - x_2)| \leq \max_{y \in [-h, h]} |(y + h)y(y - h)| \tag{17}$$

$$= \max_{y \in [-h, h]} |y^3 - yh^2| \tag{18}$$

We can maximize this expression (as shown in the lecture notes) by taking the derivative and finding that for the derivative to be 0 (a condition for a maximum):

$$3y^2 - h^2 = 0 \tag{19}$$

$$y = \pm \frac{h}{\sqrt{3}} \tag{20}$$

We can plug this value for y back into our expression above and find that the product is maximized at:

$$\frac{2}{3}\frac{1}{\sqrt{3}}h^3 \tag{21}$$

And now I can write the error and compare it to my desired accuracy:

$$e_2(\bar{x}) \leq \frac{1}{9\sqrt{3}}h^3 \leq 5 \times 10^{-7} \tag{22}$$

Solving for h we find that $h = 0.0198$ and so the number of points we need is $n = \frac{1}{h} \approx 50$.

3 Problem 3

NOTE: I used the following references to solve this problem:

(1) Dawkins, P. "Section 4-13: Newton's Method." Paul's Online Notes. `http://tutorial.math.lamar.edu/Classes/CalcI/NewtonsMethod.aspx` Visited 09/19/2019.

(2) Erdmann, Michael. "Polynomial Approximations - Interpolation." Course notes from Fall 2019.

Please see my code in code/q3.m for the full implementation of Newton's method. I used bisection as a way to obtain points in the region of convergence and then used Newton's method to obtain the final roots. The two roots that bound 7 (without any other roots in between) are 4.7122 and 7.7253, according to my algorithm. I have shown the roots on a plot of the function in Figure 3 below.

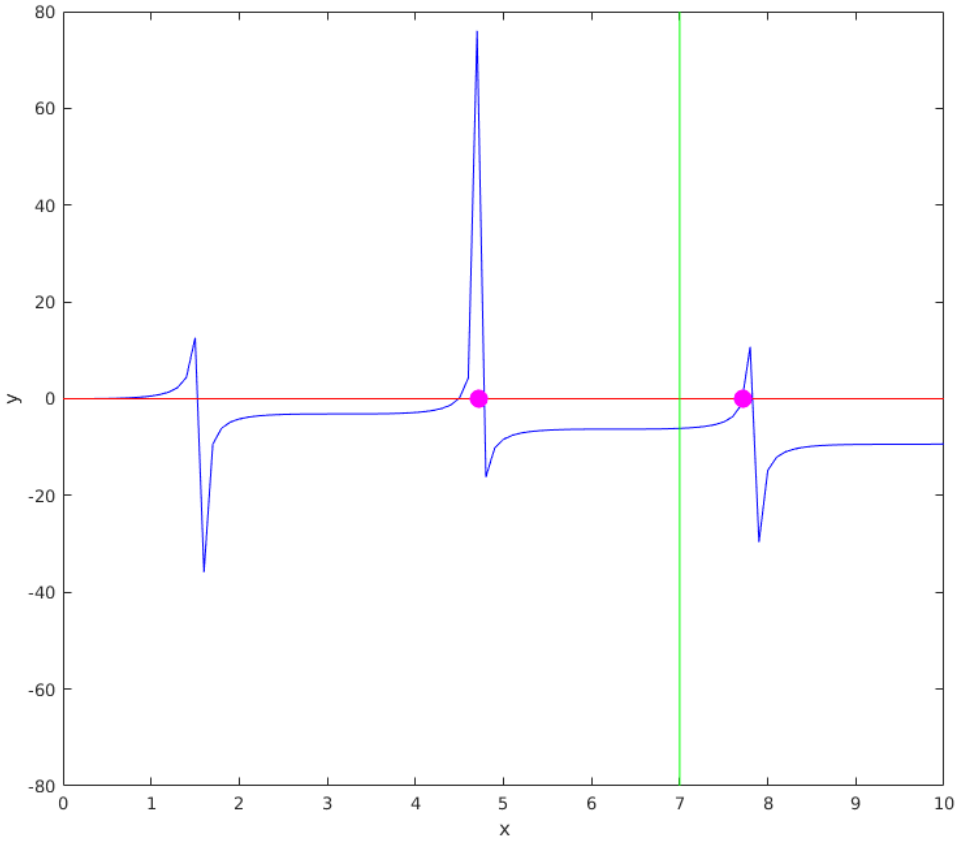

Figure 3: Plot of roots found for function $x = tanx$ using combination of bisection and Newton's method techniques.

4 Problem 4

NOTE: I used the following references to solve this problem and I had help from the TA's, thank you!

(1) Erdmann, Michael. "Polynomial Approximations - Interpolation." Course notes from Fall 2019.

If η is a root of $f(x)$ of order 2, then we can show that Newton's method no longer converges quadratically using the same logic presented in Prof. Erd-

mann's notes. Specifically, let's start with the expression for Newton's method:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{23}$$

Let's represent the second term as $h(x) = \frac{f(x_n)}{f'(x_n)}$ and write it as a Taylor series expansion. First we will substitute $x_n = \zeta + \epsilon$ because an x value that is not a root but in a region of convergence can just be written as the sum of a root and some small variation epsilon. So now we can write the Taylor series expansion of $h(\zeta + \epsilon)$ as:

$$h(\zeta + \epsilon) = h(\zeta) + \epsilon h'(\zeta) + \frac{\epsilon^2}{2} h''(\zeta) + ... \tag{24}$$

Let's look at each term in this expansion, knowing that $f(\zeta) = 0, f'(\zeta) = 0$ and $f''(\zeta) \neq 0$:

$$h(\zeta) = \frac{f(\zeta)}{f'(\zeta)} \tag{25}$$

Notice that we get an indeterminate answer for $h(\zeta)$ because it is $\frac{0}{0}$. But we can apply L'Hopital's rule and instead write $h(\zeta)$ in terms of the derivatives of the original fraction:

$$h(\zeta) = \frac{f'(\zeta)}{f''(\zeta)} = 0 \tag{26}$$

In this case $f''(\zeta) \neq 0$ so we can solve this expression. Next let's look at the derivatives of $h(\zeta)$:

$$h'(\zeta) = 1 - \frac{f(\zeta)f''(\zeta)}{[f'(\zeta)]^2} \tag{27}$$

Again, the second term is indeterminate so we need to take the derivative, and we get:

$$\frac{f(\zeta)f''(\zeta)}{[f'(\zeta)]^2} = \frac{f(\zeta)f'''(\zeta) + f'(\zeta)f''(\zeta)}{2f'(\zeta)f''(\zeta)} \tag{28}$$

Now the denominator is finite and the numerator is 0 so we can proceed with $h'(\zeta) = 1$. We use the same logic to find that $h''(\zeta) = 0$. Now we can go back to writing the Taylor series for this function:

$$h(\zeta + \epsilon) = (0) + \epsilon(1) + \frac{\epsilon^2}{2}(0) + ... \tag{29}$$

And we can plug this expansion into the original expression for Newton's method (using the substitution $x_n = \zeta + \epsilon$):

$$\zeta + \epsilon_{n+1} = \zeta + \epsilon_n - h(\zeta + \epsilon_n) = \zeta + \epsilon_n - (0 + \epsilon_n + 0 + ...) \tag{30}$$

$$\epsilon_{n+1} = 2\epsilon_n \tag{31}$$

Now we can see that the method converges linearly and C = 2.

Now if we apply a similar logic to a modified version of Newton's method we can show that it converges quadratically:

$$x_{n+1} = x_n - 2\frac{f(x_n)}{f'(x_n)} \tag{32}$$

Here we will assume that $f(\zeta) = 0; f'(\zeta) \neq 0; f''(\zeta) \neq 0; f'''(\zeta) \neq 0$ and all these derivatives are continuous in the neighborhood of ζ. This time we can write $h(\zeta)$ and its derivatives as:

$$h(\zeta) = 0 \tag{33}$$

$$h'(\zeta) = 2 \tag{34}$$

$$h''(\zeta) = 2\frac{-f''(\zeta)}{[f'(\zeta)]^3} \tag{35}$$

We can write the Taylor series expansion for $h(\zeta)$ as:

$$h(\zeta + \epsilon) = 0 + 2\epsilon + \frac{\epsilon^2}{2}\left(2\frac{-f''(\zeta)}{[f'(\zeta)]^3}\right) + ... \tag{36}$$

And plugging this into the original expression, we get:

$$\zeta - \epsilon_{n+1} = \zeta - \epsilon_n - (-2\epsilon_n + \frac{\epsilon_n^2}{2}\left(2\frac{-f''(\zeta)}{[f'(\zeta)]^3}\right) + ...) \tag{37}$$

Which results in:

$$\epsilon_{n+1} = -\epsilon_n - \frac{\epsilon_n^2}{2}\left(2\frac{-f''(\zeta)}{[f'(\zeta)]^3}\right) + ... \tag{38}$$

This should converge quadratically, I'm not sure how to get there from here...

5 Problem 5

NOTE: I used the following references to solve this problem and I had help from the TA's, thank you!

(1) Erdmann, Michael. "Polynomial Approximations - Interpolation." Course notes from Fall 2019.

(2) Mathews, J. and Fink, K. "Numerical Methods Using Matlab, 4th Edition." Prentice-Hall Inc., 2004. `http://mathfaculty.fullerton.edu/mathews/n2003/`

`mullersmethod/MullersMethodProof.pdf` Visited 09/21/2019.

5.1 Part (a)

Please see my code in code/q5.m for my implementation of Muller's method.

5.2 Part (b)

I was not able to find the correct roots to the polynomial, my function only returned one value, -7.3040. I know this value is wrong from the plot of the function. I need to do more work to debug this implementation. I do not deflate the function which may be part of my problem.

6 Problem 6

NOTE: I used the following references to solve this problem and I also got help from another student, Alex (don't know surname):

(1) Erdmann, Michael. "Polynomial Approximations - Interpolation." Course notes from Fall 2019.

6.1 Part (a)

Please see my code in code/q6.m for the "working out" of my solution. The full explanation is below. The resultant matrix is:

$$Q = \begin{pmatrix} 1 & -4 & 6 & -4 & 0 \\ 0 & 1 & -4 & 6 & -4 \\ 1 & 2 & -8 & 0 & 0 \\ 0 & 1 & 2 & -8 & 0 \\ 0 & 0 & 1 & 2 & -8 \end{pmatrix} \tag{39}$$

The determinant of Q is 0, so we know that there is a common root.

6.2 Part (b)

Please see my code in code/q6.m for the "working out" of my solution. I used the following equation to find the common root of these two polynomials:

$$\frac{x_i}{x_j} = (-1)^{i+j} \frac{det(A_i)}{det(A_j)} \tag{40}$$

Using this expression, I found that the common root was $x = 2$.

7 Problem 7

NOTE: I used the following references to solve this problem:

(1) Erdmann, Michael. "Polynomial Approximations - Interpolation." Course notes from Fall 2019.

7.1 Part (a)

My sketch of the zero contours is shown in Part (c), where I overlaid the roots on my initial sketch.

7.2 Part (b)

Please see code in code/q7.m for the "working out" of the symbolic resultant expression in x. I rearranged the given polynomials to be in terms of x as follows:

$$p(y) = 2y^2 + 4y + (3 + 2x^2 + 4x) \tag{41}$$

$$q(y) = y^2 + (2x + 5)y + (4 + x^2 + 3x) \tag{42}$$

I used these equations to write the resultant as:

$$Q = \begin{pmatrix} 2 & 4 & (3 + 2x^2 + 4x) & 0 \\ 0 & 2 & 4 & (3 + 2x^2 + 4x) \\ 1 & (2x + 5) & (4 + x^2 + 3x) & 0 \\ 0 & 1 & (2x + 5) & (4 + x^2 + 3x) \end{pmatrix} \tag{43}$$

Using MATLAB's symbolic solver (see code referenced above), I found that the determinant of Q was:

$$det(Q) = 16x^4 + 80x^3 + 144x^2 + 100x + 19 \tag{44}$$

The roots of this equation are: $-0.2979, -1.0841, -1.8090 \pm 0.636j$.

7.3 Part (c)

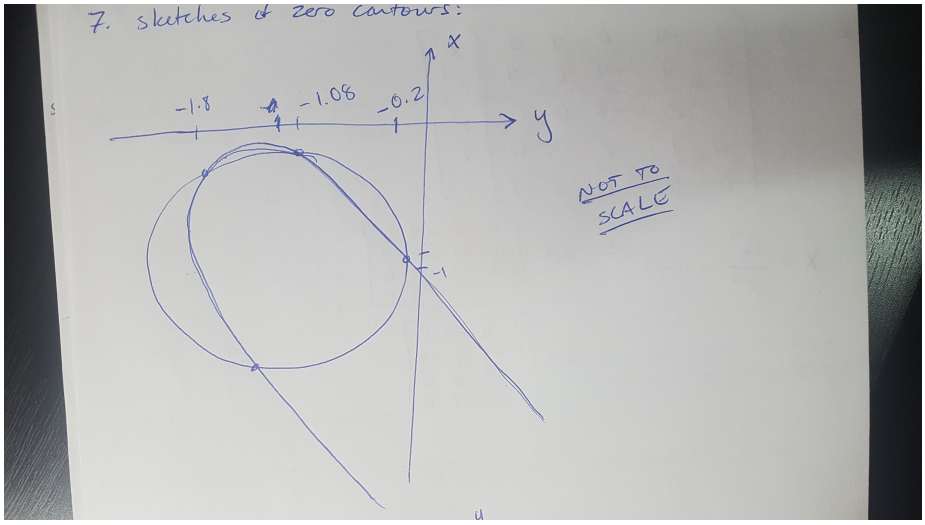

Figure 4: Hand sketch of zero contours of polynomials $p(x, y)$ and $q(x, y)$, with points of intersection from part (b) shown.

8 Problem 8

NOTE: I used the following references to solve this problem:

(1) Erdmann, Michael. "Polynomial Approximations - Interpolation." Course notes from Fall 2019.

(2) `https://www.mathworks.com/matlabcentral/answers/98665-how-do-i-plot-a-circle-with-a-give` Visited 09/22/2019.

8.1 Part (a)

Please see code in code/q8.m for the full implementation of my solution to Problem 8. I used convex combinations of points to determine whether or not a point fell inside or outside of a triangle. The basic idea of convex combinations of points is that I should be able to describe the location of any test point as the weighted sum of the three vertices of a triangle. For example, if I have a test point P_t, and a triangle with three vertices P_1, P_2, P_3, I should be able to write P_t as:

$$P_t = \alpha_1 P_1 + \alpha_2 P_2 + \alpha_3 P_3 \tag{45}$$

where there are two conditions on the coefficients, α_i:

$$\alpha_1 + \alpha_2 + \alpha_3 = 1 \tag{46}$$

and

$$0 \leq \alpha_1, \alpha_2, \alpha_3 \leq 1 \tag{47}$$

I can use these facts to write a system of linear equations that I can quickly solve in a custom MATLAB function:

$$Av = b \tag{48}$$

where

$$A = \begin{pmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{pmatrix} \tag{49}$$

$$v = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{pmatrix} \tag{50}$$

$$b = \begin{pmatrix} x_t & y_t \end{pmatrix} \tag{51}$$

By taking the inverse of A and multiplying with b, I can solve for v. I need to also check the constraint that every element in v is positive and less than 1, which I can do in an additional line of code.

8.2 Part (b)

Please see code in code/q8.m for the complete solution to this problem.

8.3 Part (c)

Choosing the 3 paths: I had two constraints on my path selection algorithm. The first was that I wanted the starting points to be within 3 units of each other in the x and y axes. I thought this would improve my chances that all 3 paths would go around the ring of fire on the same side (i.e. all three would go "north" of the ring or "south" of the ring). Since we were not given any objectives with our path planning, other than that we needed to avoid the ring and meet the rider on the other side, I did not have anything else that I wanted to optimize for.

I quickly found, though, that choosing 3 points near to each other was not enough to ensure that all 3 paths went around the ring on the same side, so I applied an additional constraint that checked if the paths were all above or below the ring at x = 5. I used interpolation to obtain a y value for each path at this point - specifically I used MATLAB's interp1() function. I only allowed the algorithm to select paths that were all on the same side of the ring.

Choosing the path weights: I think I misunderstood the problem because I choose the 3 paths I want to follow before I apply the path weights and begin to interpolate the actual path. I choose the 3 paths to follow based on their true starting points, not their weighted starting points. So in my case, the path weights help to shape the interpolated path but they make no contribution to the selection of the paths around the starting point. I played around with the weights to see how they would affect my results and without additional design criteria, I was not sure how to optimize the weights in my implementation.

For this problem (and in the plots shown below), I used $\alpha_1 = 0.5, \alpha_2 = \alpha_3 = 0.25$, but this was mostly a random selection.

Choosing the time scale: I chose to use a time scale of 0.1 in this problem. That was mostly an arbitrary decision - I wanted some decent resolution on my path but not so much that it took a long time to calculate and plot the solution. I honestly did not see any performance difference or difference in the solution when I varied the time scale from 5 down to 0.001. I am not sure what the significance of the time scale is in this problem. I can understand how it might contribute to error, but in this case I am not trying to track a function, I am trying to interpolate a new path, so I do not see how error could be a problem, unless it was so large that it caused me to run into the ring of fire.

Choosing the interpolation method: I chose to use MATLAB's interp1() function, which is a linear interpolation method (I used the default settings which are to use a linear interpolation method). I had intended to use it only for prototyping purposes and then replace it with a more refined method but it produced a correct solution when I first used it so I did not change it later.

8.4 Part (d)

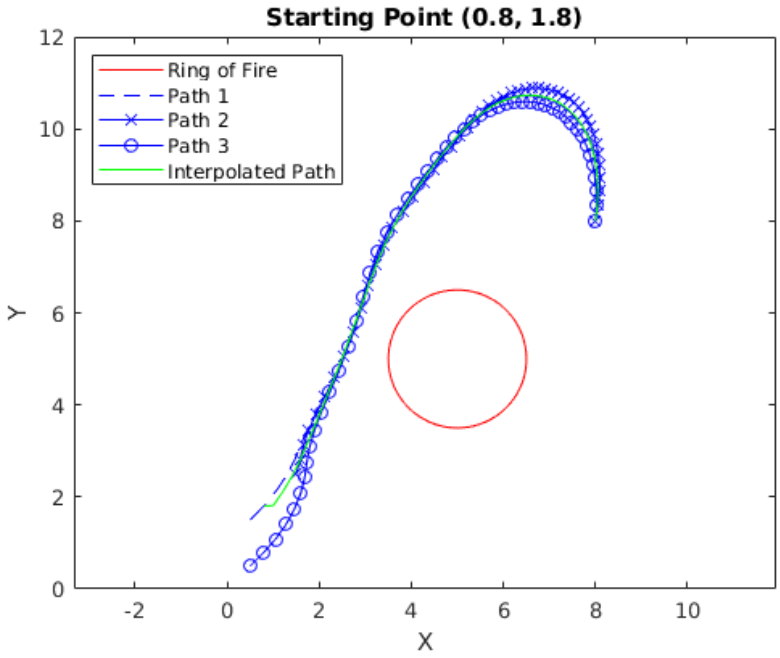

Figure 5: Solution for starting point (0.8, 1.8).

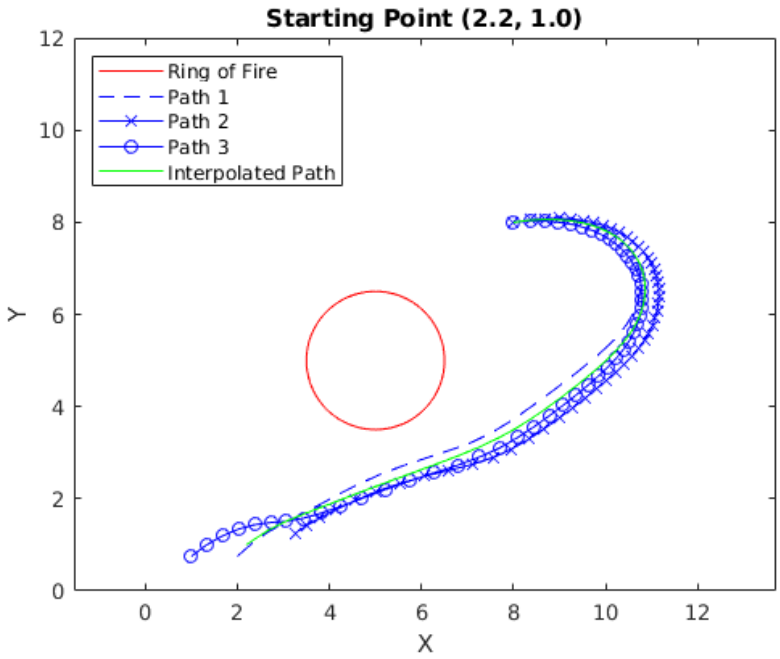

Figure 6: Solution for starting point (2.2, 1.0).

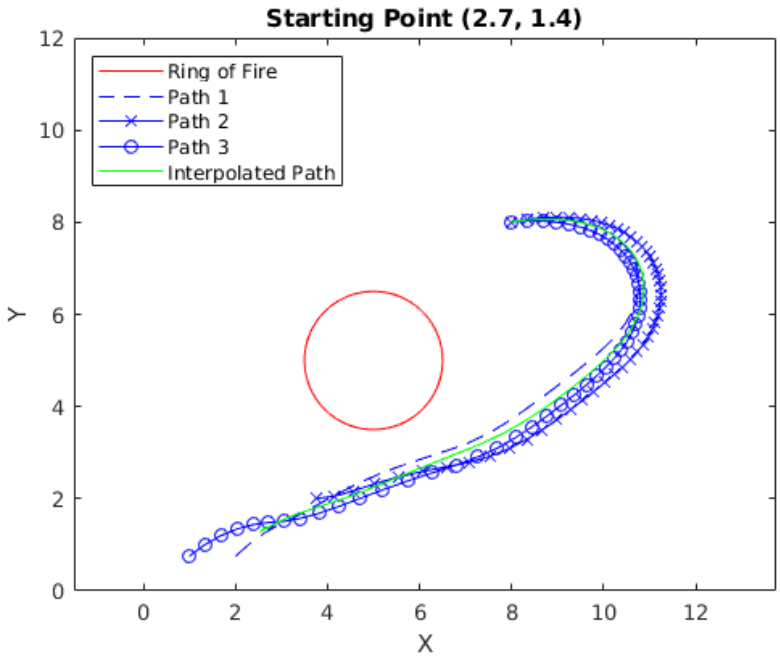

Figure 7: Solution for starting point (2.7, 1.4).

8.5 Part (e)

I would definitely need to modify my algorithm to add constraints to avoid additional obstacles. Since I hard-coded the constraint for avoiding the ring of fire, I might need to reconsider my method of avoiding obstacles and find a more generalizable solution for introducing additional obstacles. I could build additional equations into my linear system that required my interpolated path to avoid certain values. I might also find that if the paths required to avoid these additional obstacles became more intricate, that I might also need to optimize the time scale and the interpolation method more than I did in this initial implementation.