16-811 Assignment 4

Emma Benjaminson

24 October 2019

1 Problem 1

References for this problem:
(1) Drakos, Nikos, Moore, Ross, Robert. "Fourth Order Runge Kutta Method" 2002-01-28.
(2) `http://www.math.ubc.ca/~israel/m215/runge/runge.html` Visited 10/20/2019.

1.1 Part (a)

We can integrate this function as follows:

$$dy = \int \frac{1}{y} dx = log(x) + c \tag{1}$$

And given the initial condition, we can solve for the constant, c:

$$y(2) = log(2) + c = \sqrt{2} \tag{2}$$

$$c = 0.7211 \tag{3}$$

Therefore the exact analytical solution is:

$$y(x) = log(x) + 0.7211 \tag{4}$$

1.2 Part (b)

Please see my code contained in code/q1.m for the solution to this problem. The plots of both the Euler's method solution and the error over the interval [1,2] are shown in Figure 1 below. The table of solutions is also given below.

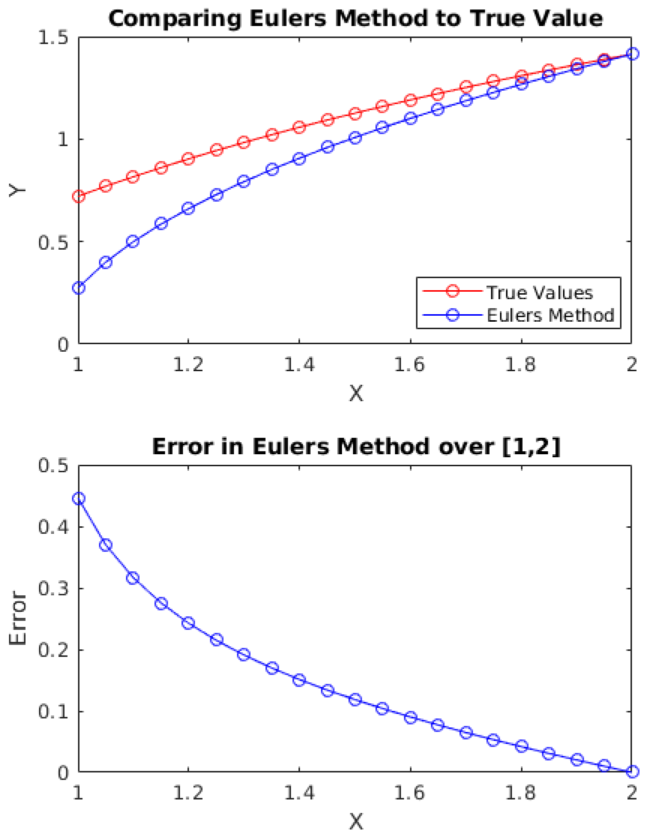

Figure 1: Plot of Euler's method over interval [1,2].

x_i	$y(x_i)$	y_i	Error
2	1.4142	1.4142	0
1.95	1.3889	1.3789	0.0101
1.9	1.363	1.3426	0.0204
1.85	1.3363	1.3054	0.0309
1.8	1.3089	1.2671	0.0418
1.75	1.2807	1.2276	0.0531
1.7	1.2517	1.1869	0.0649
1.65	1.2219	1.1447	0.0771
1.6	1.1911	1.1011	0.0901
1.55	1.1594	1.0556	0.1037
1.5	1.1266	1.0083	0.1183
1.45	1.0927	0.9587	0.134
1.4	1.0576	0.9065	0.151
1.35	1.0212	0.8514	0.1698
1.3	0.9835	0.7926	0.1908
1.25	0.9442	0.7296	0.2147
1.2	0.9034	0.661	0.2424
1.15	0.8609	0.5854	0.2755
1.1	0.8164	0.5	0.3164
1.05	0.7699	0.4	0.3699
1	0.7211	0.275	0.4461

1.3 Part (c)

Please see my code contained in code/q1.m for the solution to this problem. The plots of both the 4th order Runge-Kutta method solution and the error over the interval [1,2] are shown in Figure 2 below. The table of solutions is also given below.

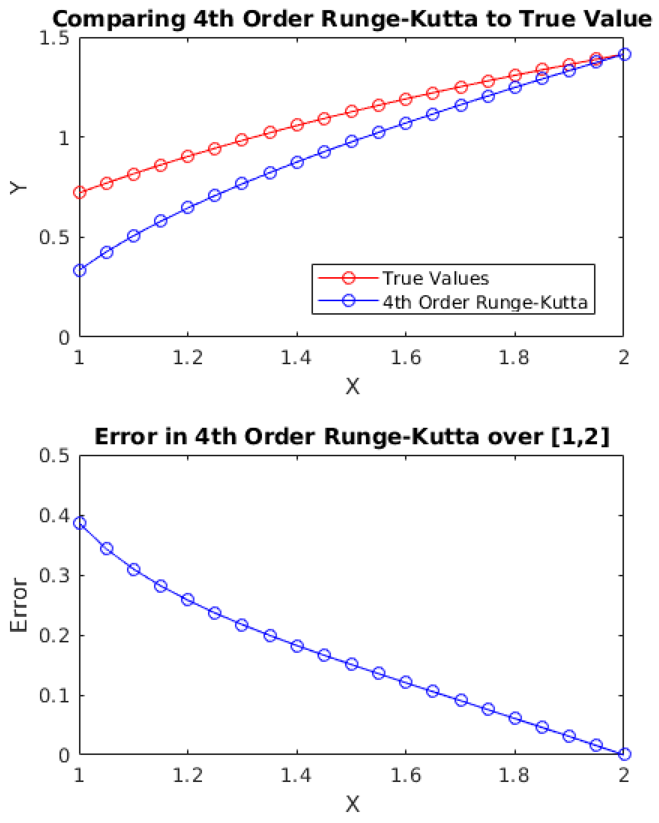

Figure 2: Plot of 4th order Runge-Kutta method over interval [1,2].

x_i	$y(x_i)$	y_i	Error
2	1.4142	1.4142	0
1.95	1.3889	1.3735	0.0154
1.9	1.363	1.3324	0.0306
1.85	1.3363	1.2906	0.0457
1.8	1.3089	1.2483	0.0606
1.75	1.2807	1.2052	0.0755
1.7	1.2517	1.1614	0.0903
1.65	1.2219	1.1167	0.1052
1.6	1.1911	1.071	0.1201
1.55	1.1594	1.0242	0.1351
1.5	1.1266	0.9761	0.1504
1.45	1.0927	0.9266	0.1661
1.4	1.0576	0.8753	0.1823
1.35	1.0212	0.822	0.1992
1.3	0.9835	0.7664	0.2171
1.25	0.9442	0.7078	0.2364
1.2	0.9034	0.6457	0.2577
1.15	0.8609	0.5792	0.2817
1.1	0.8164	0.5069	0.3096
1.05	0.7699	0.4266	0.3433
1	0.7211	0.3347	0.3864

1.4 Part (d)

Please see my code contained in code/q1.m for the solution to this problem. The plots of both the 4th order Adams Bashforth method solution and the error over the interval [1,2] are shown in Figure 3 below. The table of solutions is also given below.

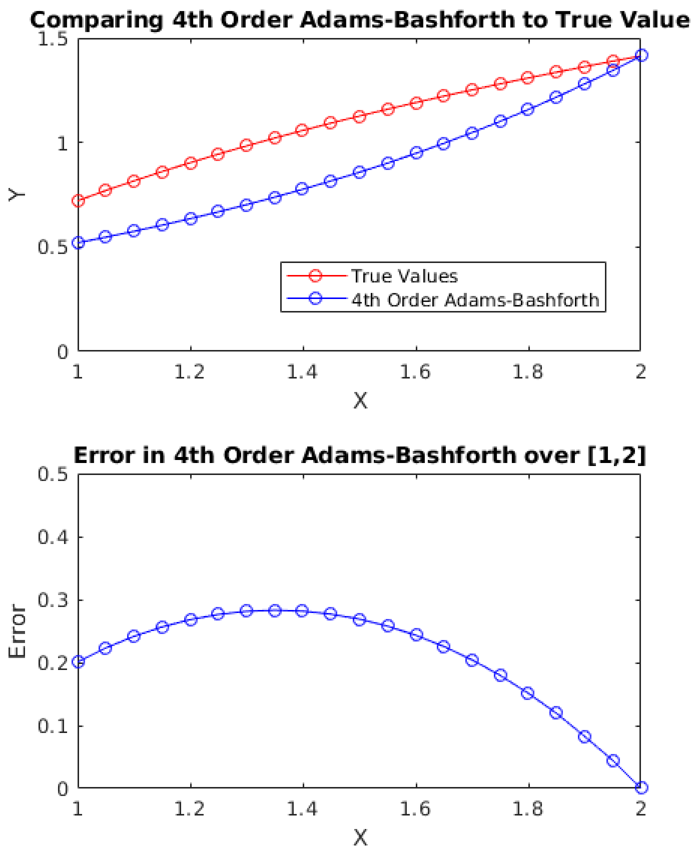

Figure 3: Plot of 4th order Adams Bashforth method over interval [1,2].

x_i	$y(x_i)$	y_i	Error
2	1.4142	1.4142	0
1.95	1.3889	1.3444	0.0445
1.9	1.363	1.2803	0.0827
1.85	1.3363	1.217	0.1193
1.8	1.3089	1.1578	0.1511
1.75	1.2807	1.1012	0.1795
1.7	1.2517	1.0476	0.2042
1.65	1.2219	0.9965	0.2254
1.6	1.1911	0.9479	0.2432
1.55	1.1594	0.9016	0.2577
1.5	1.1266	0.8577	0.2689
1.45	1.0927	0.8158	0.2768
1.4	1.0576	0.7761	0.2815
1.35	1.0212	0.7382	0.283
1.3	0.9835	0.7022	0.2813
1.25	0.9442	0.668	0.2763
1.2	0.9034	0.6354	0.268
1.15	0.8609	0.6044	0.2565
1.1	0.8164	0.5749	0.2415
1.05	0.7699	0.5469	0.223
1	0.7211	0.5202	0.2009

2 Problem 2

References for this problem:
(1) `https://en.wikipedia.org/wiki/Second_partial_derivative_test`
(2) `https://en.wikipedia.org/wiki/Hessian_matrix`

2.1 Part (a)

Please see the sketch of the isocontours in Figure 4 below.

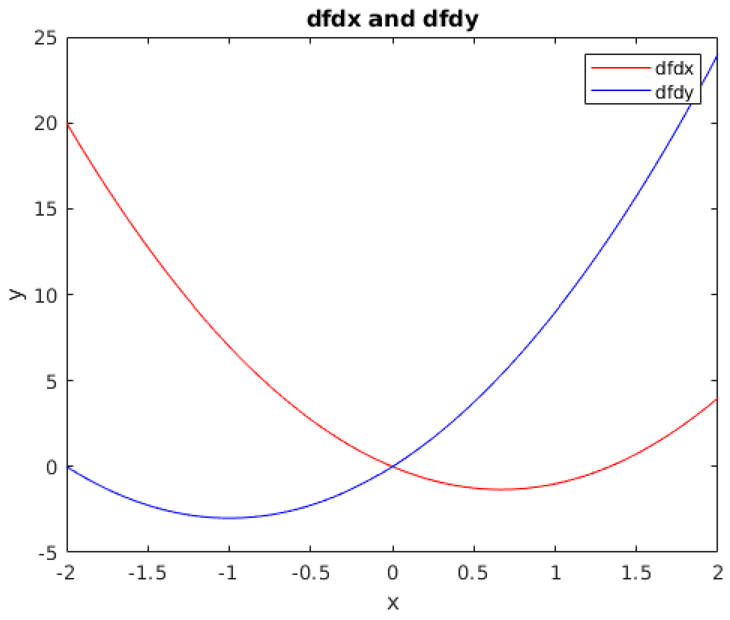

Figure 4: Plot of isocontours of f(x).

I can solve for the partial derivatives of $f(x) = x^3 + y^3 - 2x^2 + 3y^2 - 8$ and obtain the x and y components of the critical points by setting the derivatives equal to zero:

$$\frac{\partial f}{\partial x} = 3x^2 - 4x \tag{5}$$

$$\frac{\partial f}{\partial y} = 3y^2 + 6y \tag{6}$$

I find that $x_1 = 0$, $x_2 = \frac{4}{3}$ and $y_1 = 0$, $y_2 = -2$.

In order to determine if these critical points are minima, maxima or saddle points, I need to find the Hessian for the function $f(x)$ and find its determinant. According to reference (1) above:

If $|\nabla^2 f(x)| > 0$ and $\frac{\partial^2 f}{\partial x^2} > 0$, then the point is a local minimum.

If $|\nabla^2 f(x)| > 0$ and $\frac{\partial^2 f}{\partial x^2} < 0$, then the point is a local maximum.

If $|\nabla^2 f(x)| < 0$, then the point is a saddle point.

I calculate $\nabla^2 f(x)$ as follows:

$$\nabla^2 f(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix} = \begin{bmatrix} 6x-4 & 0 \\ 0 & 6y+6 \end{bmatrix} \tag{7}$$

The determinant is:

$$|\nabla^2 f(x)| = 36xy + 36x - 24y - 24 \tag{8}$$

Now I can evaluate the Hessian and its determinant at each of my four critical points. I find the following:

1. (0,0) is a saddle point
2. (0,-2) is a local maximum
3. ($\frac{4}{3}$,0) is a local minimum
4. ($\frac{4}{3}$,-2) is a saddle point

2.2 Part (b)

I will perform steepest descent on the function f by first finding $\vec{u} = \nabla f(x)$ where $x = x_0 = (1, -1)$.

$$\vec{u} = \nabla f(x_0) = \begin{bmatrix} \frac{\partial f}{\partial x}(x_0) \\ \frac{\partial f}{\partial y}(x_0) \end{bmatrix} = \begin{bmatrix} -1 \\ -3 \end{bmatrix} \tag{9}$$

Now I need to find the value of t which minimizes the expression $g(t) = f(x_0 - t\vec{u})$:

$$g(t) = f(x_0 - t\vec{u}) = 28t^3 + t^2 - 10t - 7 \tag{10}$$

The value of t that minimizes this function is $t = \frac{1}{3}$. We can plug this into the expression for x_1 as follows:

$$x_1 = x_0 - \frac{1}{3}\vec{u} = \begin{bmatrix} \frac{4}{3} \\ 0 \end{bmatrix} \tag{11}$$

We can see that $x_1 = \begin{bmatrix} \frac{4}{3} \\ 0 \end{bmatrix}$ is the local minimum critical point that we found in part (a), so we can conclude that it takes one step in the steepest descent algorithm to reach an overall local minimum of f.

3 Problem 3

References for this problem:
(1) http://www.math.hawaii.edu/ lee/linear/eigen.pdf

3.1 Part (a)

Let us denote two eigenvectors as x_i and x_j, and let us assume they have distinct eigenvalues λ_i and λ_j. We also know that the matrix Q is symmetric. We are interested in whether or not these two eigenvectors can be proven to have Q-orthogonality, so let's start by writing out the expression for Q-orthogonality which I will need to prove is true:

$$x_i^T Q x_j \tag{12}$$

Notice that this expression is equal to:

$$x_i^T Q x_j = (x_j^T Q x_i)^T \tag{13}$$

From the definition of an eigenvector, I can write:

$$Q x_{i,j} = \lambda_{i,j} x_{i,j} \tag{14}$$

Let's substitute this into the equation above:

$$x_i^T \lambda_j x_j = (x_j^T \lambda_i x_i)^T \tag{15}$$

Now let's evaluate the transpose (notice that λ is a scalar so the transpose of it is just itself):

$$x_i^T \lambda_j x_j = x_i^T \lambda_i x_j \tag{16}$$

Now let's move all the terms to one side:

$$x_i^T x_j (\lambda_j - \lambda_i) = 0 \tag{17}$$

This tells us that if the eigenvalues are distinct, then the only way that this expression is true is if x_i and x_j are orthogonal to one another. Therefore, any two eigenvectors of Q with distinct eigenvalues are Q-orthogonal.

3.2 Part (b)

If we know that any two eigenvectors can be shown to be pairwise orthogonal, then we know that the product $x_i^T x_j = 0$ always so any of these two basis vectors are Q-orthogonal as well.

4 Problem 4

4.1 Part (a)

We can begin with the definition of $d_k = -g_k + \beta_{k-1} d_{k-1}$:

$$d_k^T Q d_k = d_k^T Q(-g_k + \beta_{k-1} d_{k-1}) = -d_k^T Q g_k + d_k^T Q \beta_{k-1} d_{k-1} \tag{18}$$

We can substitute in the expression for β_{k-1}:

$$d_k^T Q d_k = -d_k^T Q g_k + d_k^T Q \left(\frac{g_k^T Q d_{k-1}}{d_{k-1}^T Q d_{k-1}} \right) d_{k-1} \tag{19}$$

We know that $d_k^T Q d_i = 0$ for $i < k$ so we can set the second term equal to 0 and thus we are left with:

$$d_k^T Q d_k = -d_k^T Q g_k \tag{20}$$

Given this fact, I can show that when I solve for x_{k+1}, I only need Q for evaluating g_k and Qg_k:

$$x_{k+1} = x_k + \alpha_k d_k \tag{21}$$

α_k is defined as:

$$\alpha_k = \frac{-g_k^T d_k}{d_k^T Q d_k} = \frac{g_k^T d_k}{d_k Q g_k} \tag{22}$$

4.2 Part (b)

Now I want to evaluate Qg_k and show that it is equal to $g_k - p_k$, which I can do using two facts: $g_k = Qx_k + b$ and $p_k = \nabla f(y_k) = Qy_k + b$ where $y_k = x_k - g_k$:

$$Qg_k = g_k - p_k = Qx_k + b - (\nabla f(y_k)) = Qx_k + b - \big(Qy_k + b\big) = Qx_k + b - \big(Q(x_k - g_k) + b\big) \tag{23}$$

Now rearranging terms I find:

$$Qg_k = Qx_k + b - Qx_k + Qg_k - b = Qg_k \tag{24}$$

Therefore we know that $Qg_k = g_k - p_k$.

4.3 Part (c)

Now we will derive the conjugate gradient algorithm without requiring knowledge of the Hessian of the function, f (i.e. without needing to know Q). The steps of the algorithm are:

1. $d_0 = -g_0$
2. $\alpha_k = \frac{-g_k^T d_k}{d_k^T Q d_k} = \frac{g_k^T d_k}{d_k^T Q g_k} = \frac{g_k^T d_k}{d_k^T (g_k - p_k)}$
3. $x_{k+1} = x_k + \alpha_k d_k$
4. $\beta_k = \frac{g_{k+1}^T Q d_k}{d_k^T Q d_k}$ Note that β_k is a scalar so we can take the transpose and still get the same value. Therefore: $\beta_k = \frac{d_k^T Q g_{k+1}}{d_k^T Q d_k} = \frac{d_k^T Q g_{k+1}}{-d_k^T Q g_k} = \frac{d_k^T (g_{k+1} - p_{k-1})}{-d_k^T (g_k - p_k)}$
5. $d_{k+1} = -g_{k+1} + \beta_k d_k$
6. Return x_n

5 Problem 5

References for this problem:
(1) http://www.ai.mit.edu/courses/6.891-nlp/lagrange.pdf

We can define the function $f(x, y) = xy$ to describe the area of the figure that we are trying to maximize. We can write the constraint as $h(x, y) = 2x + 2y - p$ where p is the fixed perimeter of the figure. Now we can use the method of Lagrangian multipliers to find the length and width of the rectangle with the greatest area for a fixed perimeter:

$$F(x, y, \lambda) = f(x, y) + \lambda h(x, y) = xy + \lambda(2x + 2y - p) \tag{25}$$

We take the gradient of $F(x, y, \lambda)$:

$$\nabla F = \begin{bmatrix} \frac{\partial f}{\partial x} + \lambda\frac{\partial h}{\partial x} \\ \frac{\partial f}{\partial y} + \lambda\frac{\partial h}{\partial y} \\ h \end{bmatrix} = \begin{bmatrix} y + \lambda(2) \\ x + \lambda(2) \\ 2x + 2y - p \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{26}$$

Now we have a system of 3 equations that we can solve to find x and y in terms of p, as follows. Given that $y = -2\lambda$ and $x = 2\lambda$, we can solve for $\lambda = \frac{-p}{8}$. We can substitute this value back into the expressions for x and y and find that $y = x = \frac{p}{4}$. In other words, the rectangle with the most area for a given perimeter is a square.

Now let's check the second order sufficiency conditions. The first condition states that: $\exists x^*$ s.t. $h(x^*) = 0$ and $\exists \lambda \in \mathbb{R}$ s.t. $\nabla f(x^*) + \lambda^T \nabla h(x^*) = 0$. If we assume that $x^* = \begin{bmatrix} \frac{p}{4} \\ \frac{p}{4} \end{bmatrix}$ and $\lambda = \frac{-p}{8}$ then:

$$\begin{bmatrix} \frac{p}{4} \\ \frac{p}{4} \end{bmatrix} + \left(\frac{-p}{8}\right) \begin{bmatrix} 2 \\ 2 \end{bmatrix} = 0 \tag{27}$$

Therefore the first condition has been met. The second condition is $L(x^*) = \nabla^2 f(x^*) + \lambda^T \nabla^2 h(x^*)$ is positive definite. Unfortunately I found that $\nabla^2 f(x^*) = \nabla^2 h(x^*) = 0$ so I am not sure that this condition is met.

6 Problem 6

References for this problem:
(1) `https://en.wikipedia.org/wiki/Gradient_descent`

6.1 Part (a)

Please see the code contained in code/q6.m for this problem. Figure 5 shows a plot of my path after one iteration. Please note that I scaled my negative gradient by 5 (i.e. gamma = 5 in my code) instead of by 0.1 because it better illustrated that my path was slowly moving away from the obstacles.

Figure 6 shows the path after convergence when I continued to scale my gradient by 5. When I reduced the scaling to 0.1 as described in the problem, I noticed that my path points were moving closer together as in Figure 7. I realize this is probably wrong because the question asks us to keep the start and end points of the path fixed. I am not sure what I am doing wrong in my code implementation.

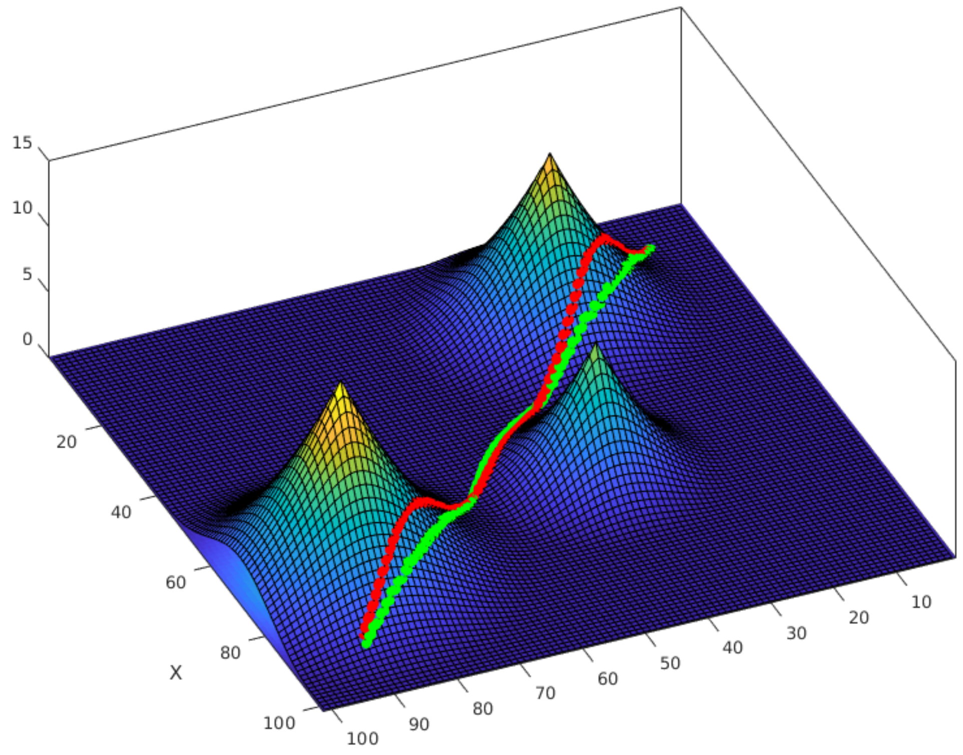

Figure 5: Plot my path (green) against initial path (red) after one iteration, $\gamma = 5$.

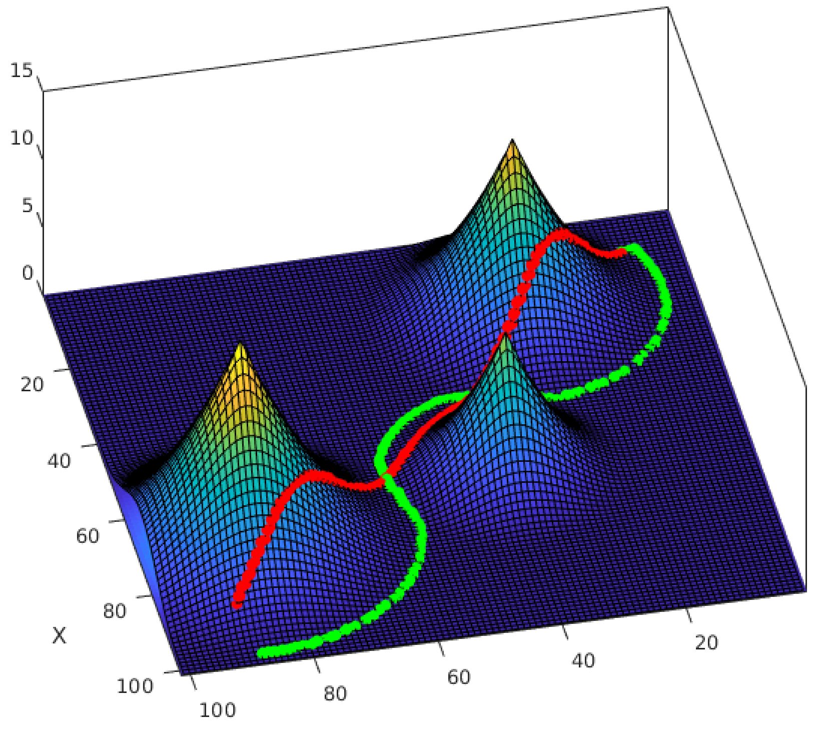

Figure 6: Plot my path (green) against initial path (red) after 100 iterations, $\gamma = 5$.

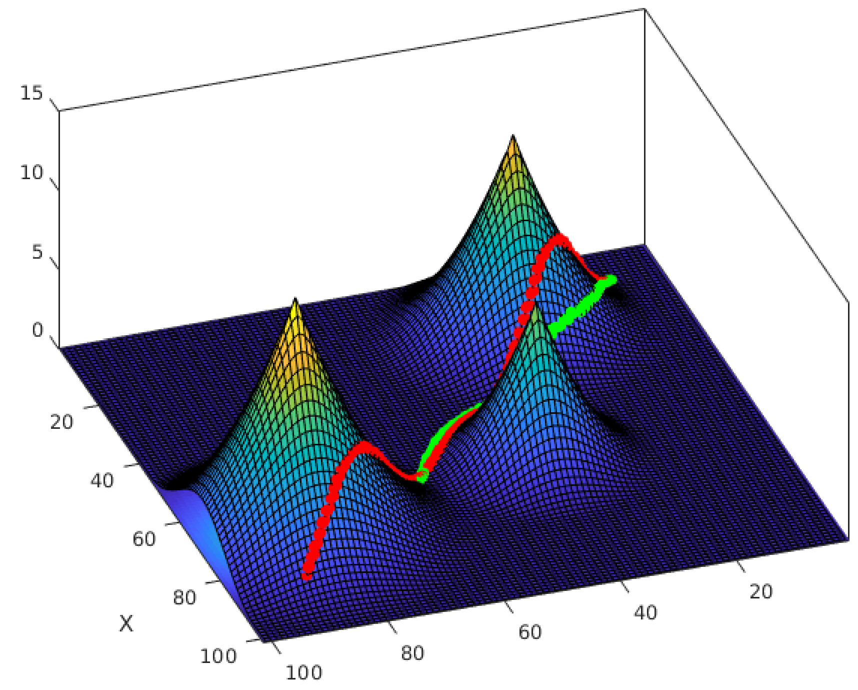

Figure 7: Plot my path (green) against initial path (red) after 100 iterations, $\gamma = 0.1$.

6.2 Part (b)

Please see the code contained in code/q6.m for this problem. I am not sure that my implementation is correct because the path values all converge to 0 (see Figure 8 below) and spacing my points correctly was never an issue that I saw in Part a. I have tried to implementing a smoothing cost in my code but it appears to be doing something wrong.

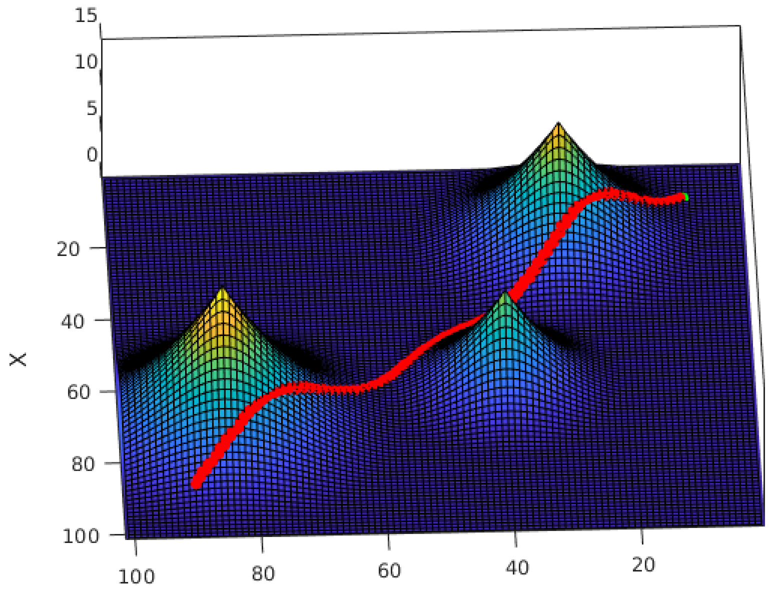

Figure 8: Plot my path (green) against initial path (red) after 100 iterations.

6.3 Part (c)

Please see the code contained in code/q6.m for this problem. Again, I did not see a spacing issue with my points in my initial implementation and my smoothness cost function generated a trajectory that went in the reverse direction, see Figure 9. I am not sure what I am doing wrong.

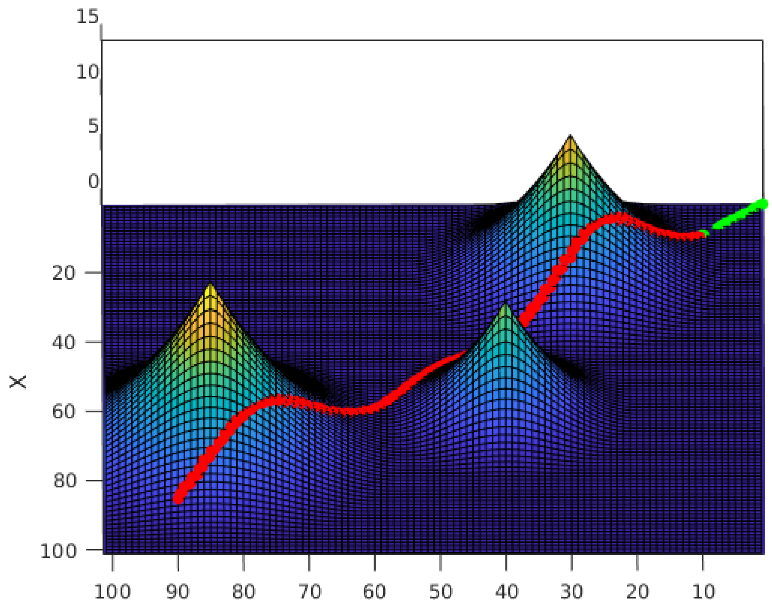

Figure 9: Plot my path (green) against initial path (red) after 100 iterations.

6.4 Part (d)

Starting from different trajectories will produce different final trajectories. The gradient descent algorithm is trying to minimize the cost function, but the cost function could have multiple local minima where the gradient descent algorithm could end up. Considered another way, the initial path for a given starting point may traverse a ”hill” on one side, and the gradient descent will take the path down to that side of the hill. But if the initial path traversed a hill on the other side with a different starting point, the gradient descent algorithm would likely take the path down to that opposite side of the hill.

6.5 Part (e)

One strategy would be to write another cost function to prevent the robot from entering these areas. Another would be to divide up the path into subsections, and use the algorithm to traverse sections of the path, and each subsection terminates away from these high cost areas so that you can force the algorithm to avoid them by setting starting and ending conditions that prevent the robot from entering these areas.