MTH-308B Mini Project 1

Aniket Pandey

Roll No: 160113

Problem Statement

Given a parametric curve C in $\mathbb{R}^2$ given by $C : [0,1] \rightarrow \mathbb{R}^2$, such that

$$C(t) = (X(t), Y(t)), 0 \leq t \leq 1 \tag{1}$$

and a point (x_0, y_0), we need to find the orthogonal projection of this point on the curve. The orthogonal projection from (x_0, y_0) on $C(t)$ will be the point on the curve closest to it.

Proposed Algorithm

The method used for the purpose of numerically solving this problem is a variant of the **Steepest Descent algorithm** and **Newton-Raphson** (secant) method.
The following steps describe the algorithm used -

1. Since the period of the given curve is 1,

$$f(t+1) = f(t) \tag{2}$$

   The distance squared function is given by -

$$f(t) = (X(t) - x_0)^2 + (Y(t) - y_0)^2 \tag{3}$$

   A period of 1 implies that the possible values of t lie in the range $[0,1]$. After this range, the behavior of the function will repeat.
   Considering various possibilities in *epsValue*, the curve is divided into 200 sections and hence, the values of t are iterated from 0.0 to 0.995 with a step value of 0.005.

2. By gradient descent algorithm, the value of the parameter t at $(n+1)^{th}$ iteration is given by

$$t_{n+1} = t_n - k\nabla f(t_n) \tag{4}$$

   The above equation is dependent on k. An ideal value of k will be the one which minimizes the given parametric function

$$f(t_{n+1}) = f(t_n - k\nabla f(t)) \tag{5}$$

   $\Rightarrow$ k satisfies the following equation

$$\nabla f(t_{n+1}).\nabla f(t_n) = 0 \tag{6}$$

3. Ideally in this case, we need to find the roots of the problem using a numerical method.

   One approach for doing so will be to use Matlab's symbolic toolbox. However, this computation will add up to the total runtime.

   Another approach for this will be to use a numerical method to solve for the roots. However, this will reduce the probability that the convergence criterion is met and the solution may diverge.

4. It should be noted that we initially computed the value of t which gives a minimum value of distance function. Post which we need to use *Newton-Raphson* (secant) method for bringing t_0 within the desired accuracy. The ϵ is defined as:

$$cos(f(t))(eps) = \frac{[X(t_c) - X_0, Y(t_c) - Y_0] \cdot [X'(t_c), Y'(t_c)]}{\|[X(t_c) - X_0, Y(t_c) - Y_0]\|_2 \, \|[X'(t_c), Y'(t_c)]\|_2} \tag{7}$$

5. Suppose the root of the problem is such that initial guess (X_t, Y_t) makes an *obtuse* angle with $(X_c - X_t, Y_c - Y_t)$. In this case the gradient of the curve is negative at t.

   Let $a = 0.001$ be arbitrarily chosen. We use $a - \epsilon$ as the variable step for our Steepest descent method. Now,

$$|a - \epsilon| > 0 \Rightarrow |a - \epsilon| \cos(f(t)) < 0 \tag{8}$$

   Therefore, in the n^{th} iteration, the computed value of t_{n+1} increases (i.e. more than the previous value t_n, converging towards the solution.

$$\nabla t_n = t_{n+1} - t_n \approx -(\epsilon/2)\cos(f(t)) > 0 \tag{9}$$

   Without loss of generality, the same can be proven for the case when the solution lies to the right of our initial guess. The computed value of t_{n+1} will move towards left again, converging towards the solution.

   Hence, we make a gradual progress converging towards the solution at each iteration.

6. Since the problem statement does not emphasize of maximizing the efficiency. We can relax the restriction on the number of Newton-Raphson iterations. However, to avoid an infinite loop in case of a malicious input or a badly conditioned function, we limit the maximum number of iterations to **100000**[1].

---

[1]The *step* value in NR is so chosen as to heavily-optimize the algorithm for the case when $\epsilon \approx 10^{-14}$, hinted as a possible value for ϵ by Prof. Akash Anand. Hence, it is reasonable for the algorithm to not be efficient for relatively larger values of ϵ. Thus to circumvent the worst case possibility, iteration limit was chosen to be a very large number, i.e 100000 instead of say, 5000.