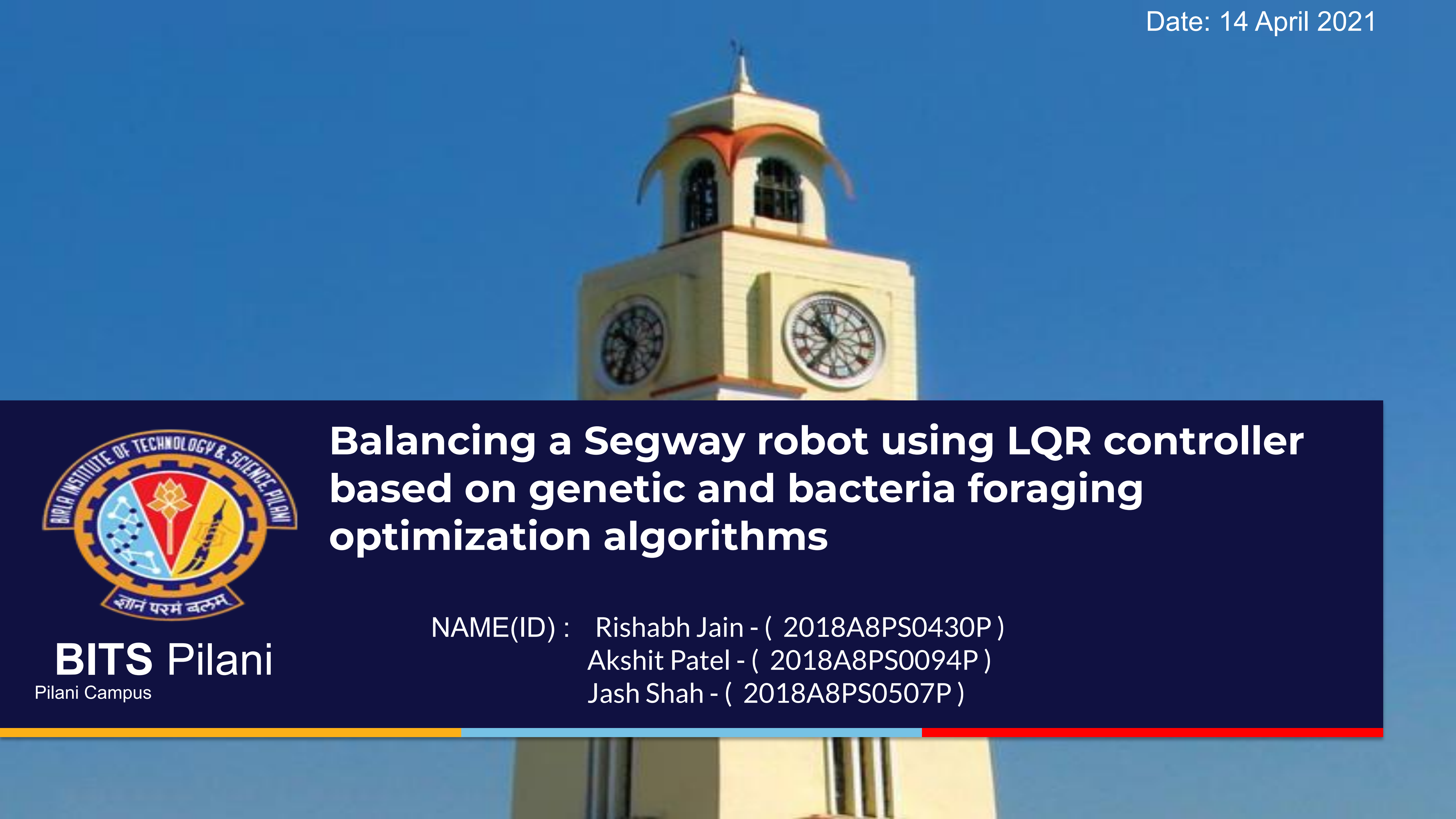
Date: 14 April 2021



BITS Pilani
Pilani Campus

Balancing a Segway robot using LQR controller based on genetic and bacteria foraging optimization algorithms

NAME(ID) : Rishabh Jain - (2018A8PS0430P)
Akshit Patel - (2018A8PS0094P)
Jash Shah - (2018A8PS0507P)

Aim

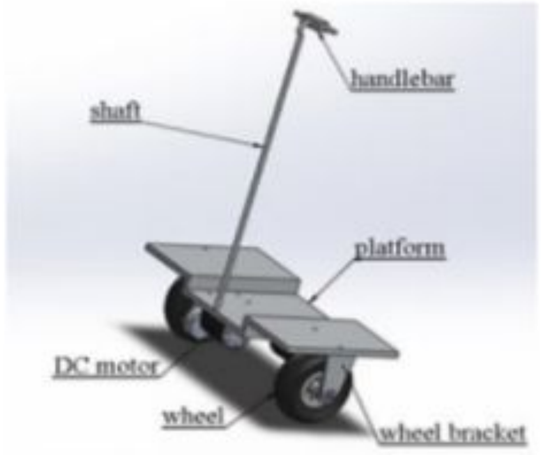
Our aim of the project is to design optimal control of the segway robot using lqr and soft computing optimization techniques like genetic algorithm and bacteria foraging algorithm.

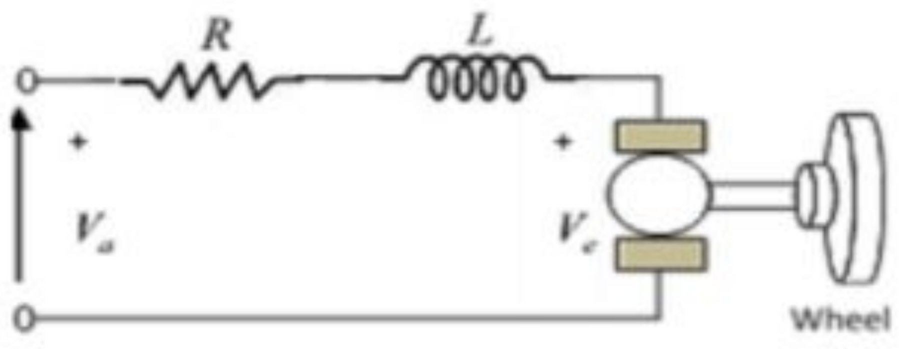
Introduction

The Segway robot is an electric, two-wheeled self-balancing human transporter with a computer-controlled gyroscopic stabilization and control system.The mathematical model of the system dynamics is derived and then state-space formulation for the system is presented to enable the design of a state feedback controller scheme. In this research, an optimal control system based on the linear quadratic regulator (LQR) technique is proposed to stabilize the mobile robot.Two tuning methods, genetic algorithm (GA) and bacteria foraging optimization algorithm (BFOA) are used to obtain optimal values for controller parameters.

Description of the Robot

The Segway is an extraordinary robot with very little input constraint and a high degree of freedom. This makes it very convenient and fast for use even for even law enforcement. A good amount of control is required to balance the vehicle as it is a combination of a **Unicycle and an Inverted Pendulum. It now becomes very intuitive that the inverted pendulum needs to be balanced**





Modelling of Segway Robot

$$\ddot{\phi} = \frac{M_p I}{(I_p + M_p I^2)}\ddot{x} + \frac{2K_m K_e}{Rr(I_p + M_p I^2)}\dot{x} + \frac{M_p g I}{Rr(I_p + M_p I^2)}\phi - \frac{2K_m}{(I_p + M_p I^2)}V_a$$

$$\ddot{x} = \frac{2K_m}{RrK_w}V_a - \frac{2K_m K_e}{Rr^2 K_w}\dot{x} + \frac{M_p I}{Rr^2 K_w}\ddot{\phi}$$

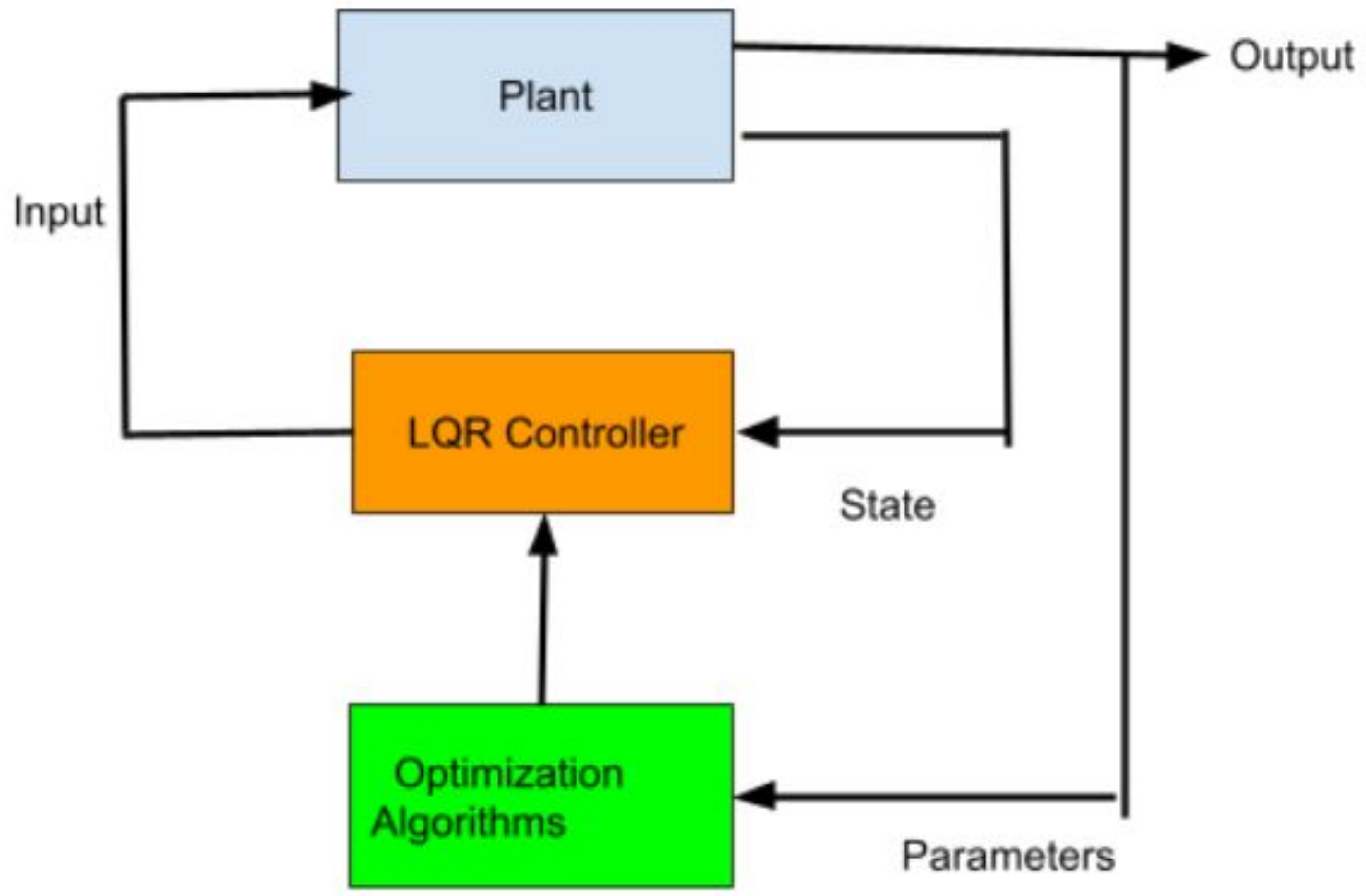
where, $K_w = 2M_w + 2I_w/r$

State Space Equation

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\phi} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{K_1(M_p I r - I_p - M_p I^2)}{Rr^2 \alpha} & \frac{M^2{}_p g I^2}{\alpha} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{K_1(r\beta - M_p I)}{Rr^2 \alpha} & \frac{M_p g I \beta}{\alpha} & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \phi \\ \dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{2K_m(I_p + M_p I^2 - M_p I r)}{Rr\alpha} \\ 0 \\ \frac{2K_m(M_p I - \beta)}{Rr\alpha} \end{bmatrix} V_a$$

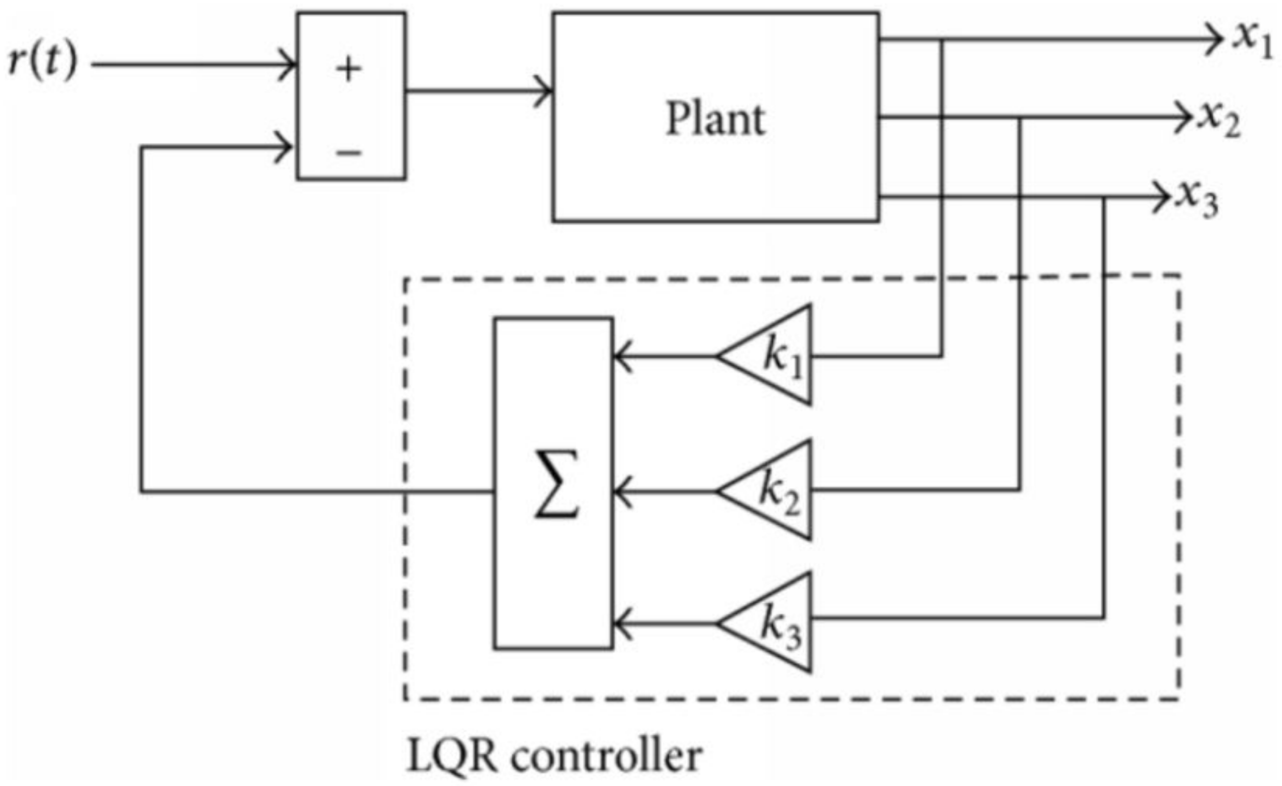
where, $K_1 = 2K_m K_e, \beta = 2M_w + \frac{2I_w}{r^2} + M_p$ and $\alpha = \left[I_p \beta + 2M_p I^2 (M_w + \frac{I_w}{r^2})\right]$

Methodology



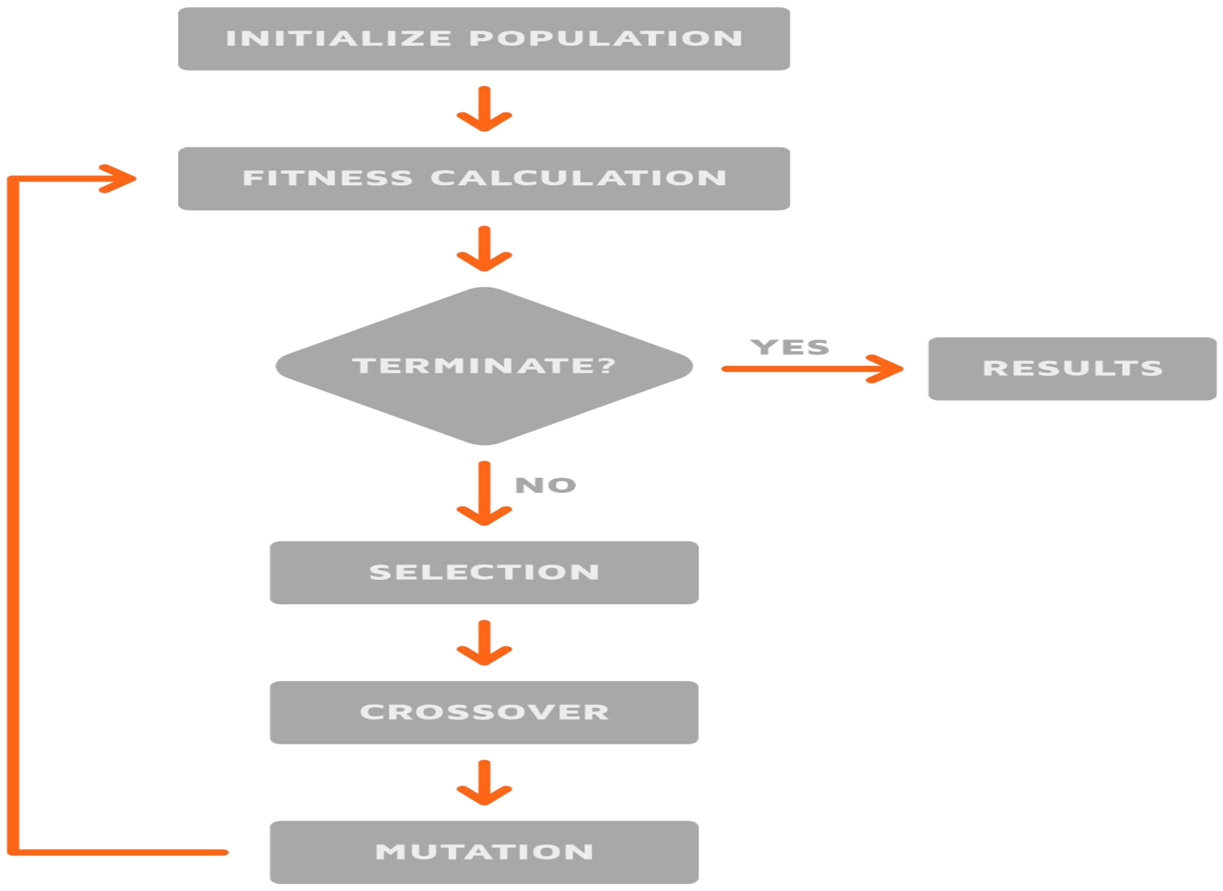
LQR

The Linear Quadratic Regulator (**LQR**) is a well-known method that provides optimally controlled feedback gains to enable the closed-loop stable and high performance design of systems.The **LQR** algorithm reduces the amount of **work** done by the control systems engineer to optimize the controller.The **LQR** algorithm is essentially an automated way of finding an appropriate state-feedback controller.



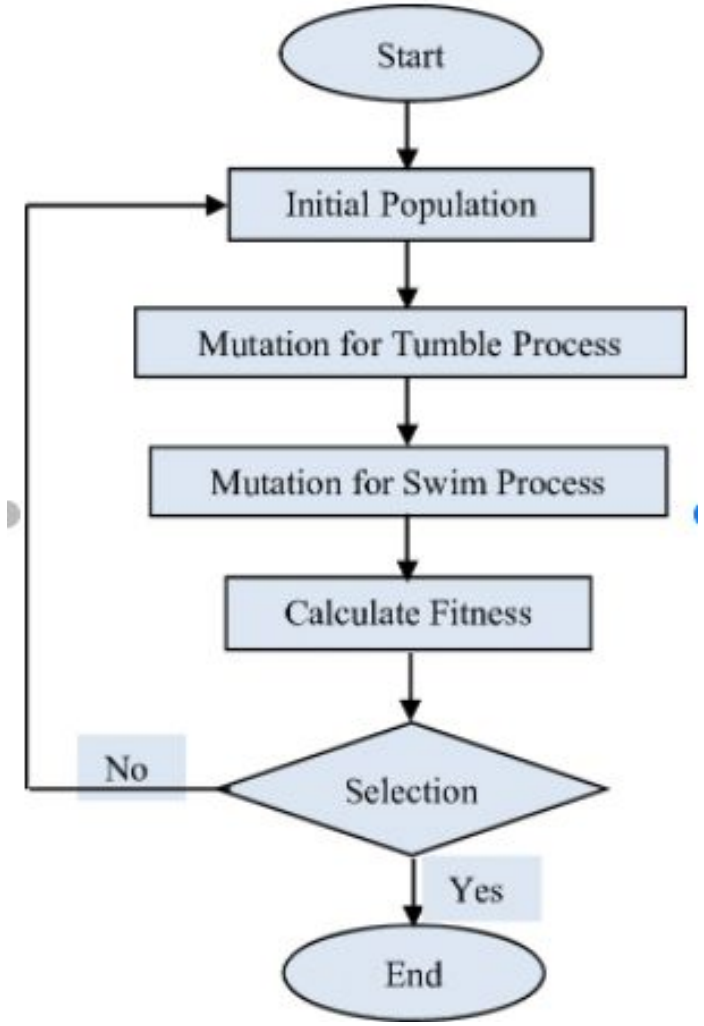
Genetic Algorithm

Genetic algorithm is a method based on reproduction of DNA in various generations in living beings. We try to simulate the process for finding an optimized solution.The pseudo code for the process is shown in the figure.



BFOA

Forging strategies are methods for locating, handling, and ingesting food. This method favours those bacterias that have successful foraging strategies or the bacterias with enough food s.t they can reproduce.After many generations, poor foraging strategies are either eliminated or shaped into good ones i.e. redesigned.



Fitness Function

Fitness function in the paper has not been defined properly, first, we took fitness function to be variation from desired values which is mentioned in the paper but as rising time and settling time were not varying much with each population, only small changes in overshoot we directly added all three values which seemed to provide better results. So our fitness function was the addition of rising time, settling time, and overshoot of position vs time graph in each iteration.

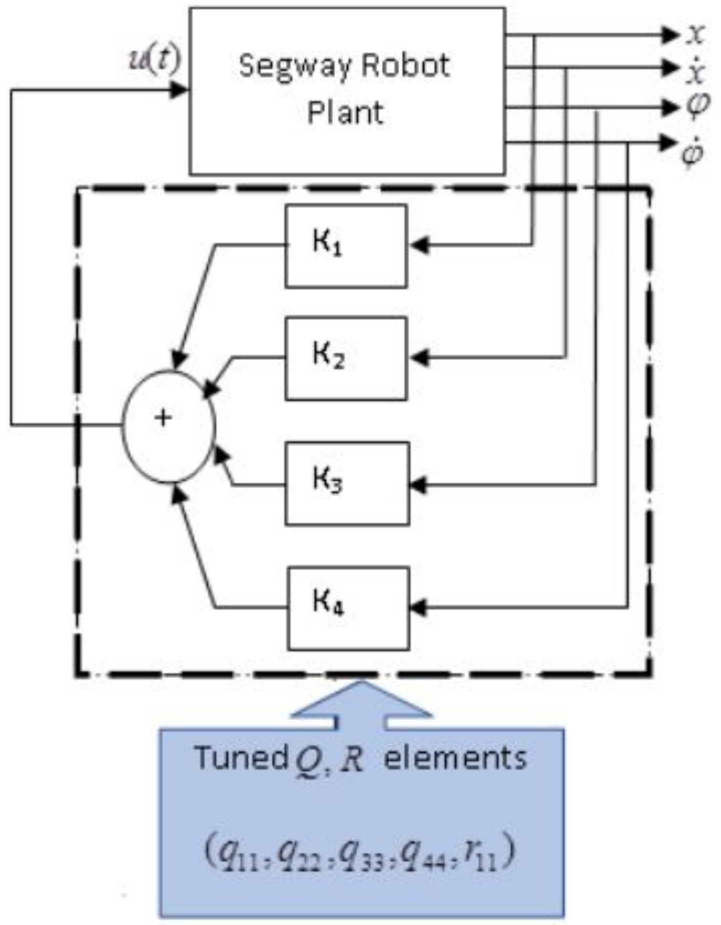
Fitness Function = |Rise Time| + |Settling Time| + |Overshoot |

MATLAB Simulation

Model Used for Simulation

$$\begin{bmatrix}\dot{x}\\ \ddot{x}\\ \dot{\phi}\\ \ddot{\phi}\end{bmatrix} = \begin{bmatrix}0 & 1 & 0 & 0\\ 0 & -0.0316 & 0.3817 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0.3927 & 49.5531 & 0\end{bmatrix}\begin{bmatrix}x\\ \dot{x}\\ \phi\\ \dot{\phi}\end{bmatrix} + \begin{bmatrix}0\\ 0.05746\\ 0\\ -0.7141\end{bmatrix}V_a$$

$$y = \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix}x\\ \dot{x}\\ \phi\\ \dot{\phi}\end{bmatrix} + \begin{bmatrix}0\\ 0\end{bmatrix}u$$



Genetic LQR Parameters

Population Size = 15

Generation = 10

Best Q Obtained =

q11=7.056311630519287

q22=0.040608399663108

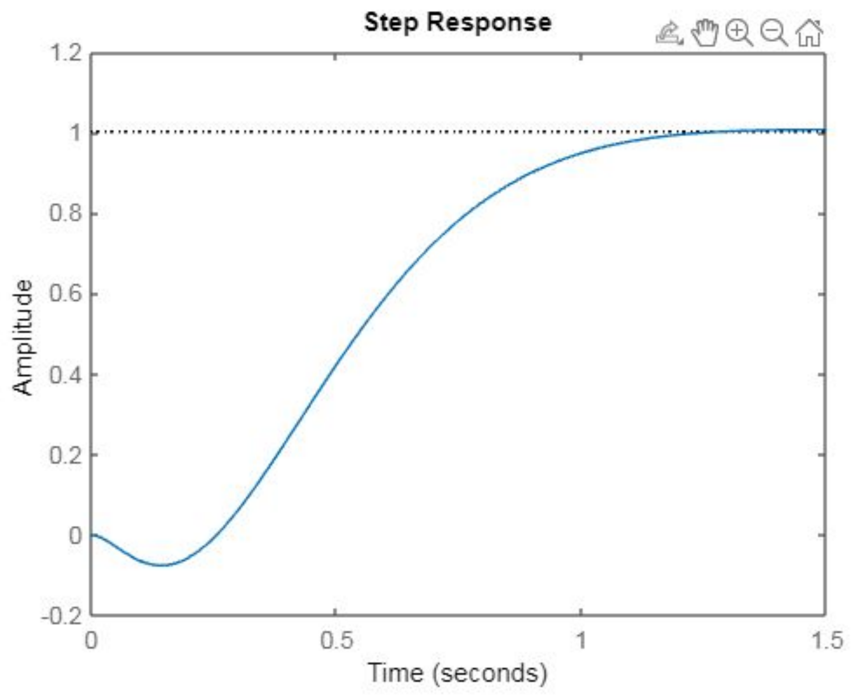
q33=0.150562417774016

q44=0.030452791998497

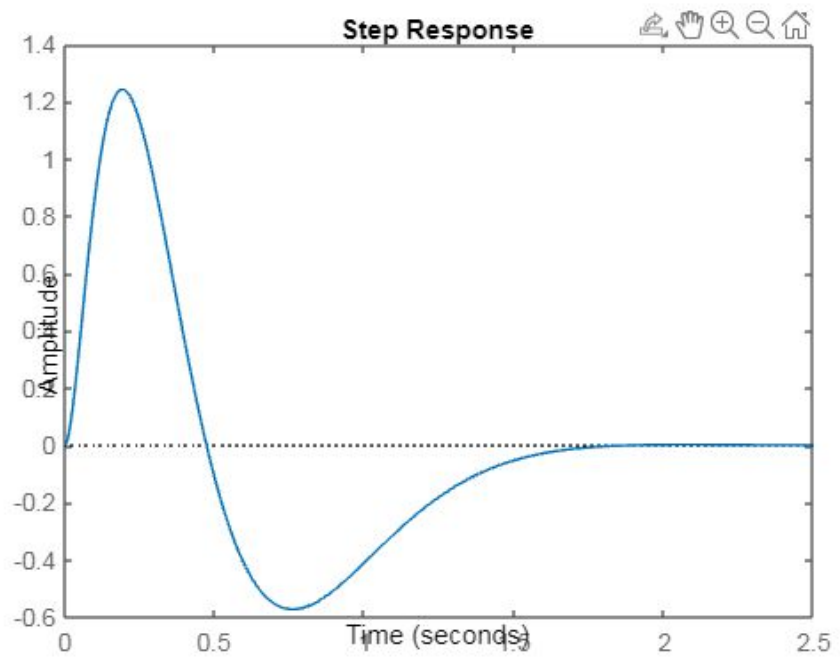
Best R obtained = 0.000016648572929

K = 1.0e+02 *[-6.510290527654438 -3.928285136135124 -5.775516991619111 -0.891476241249279]

GA graphs(step response vs amplitude)

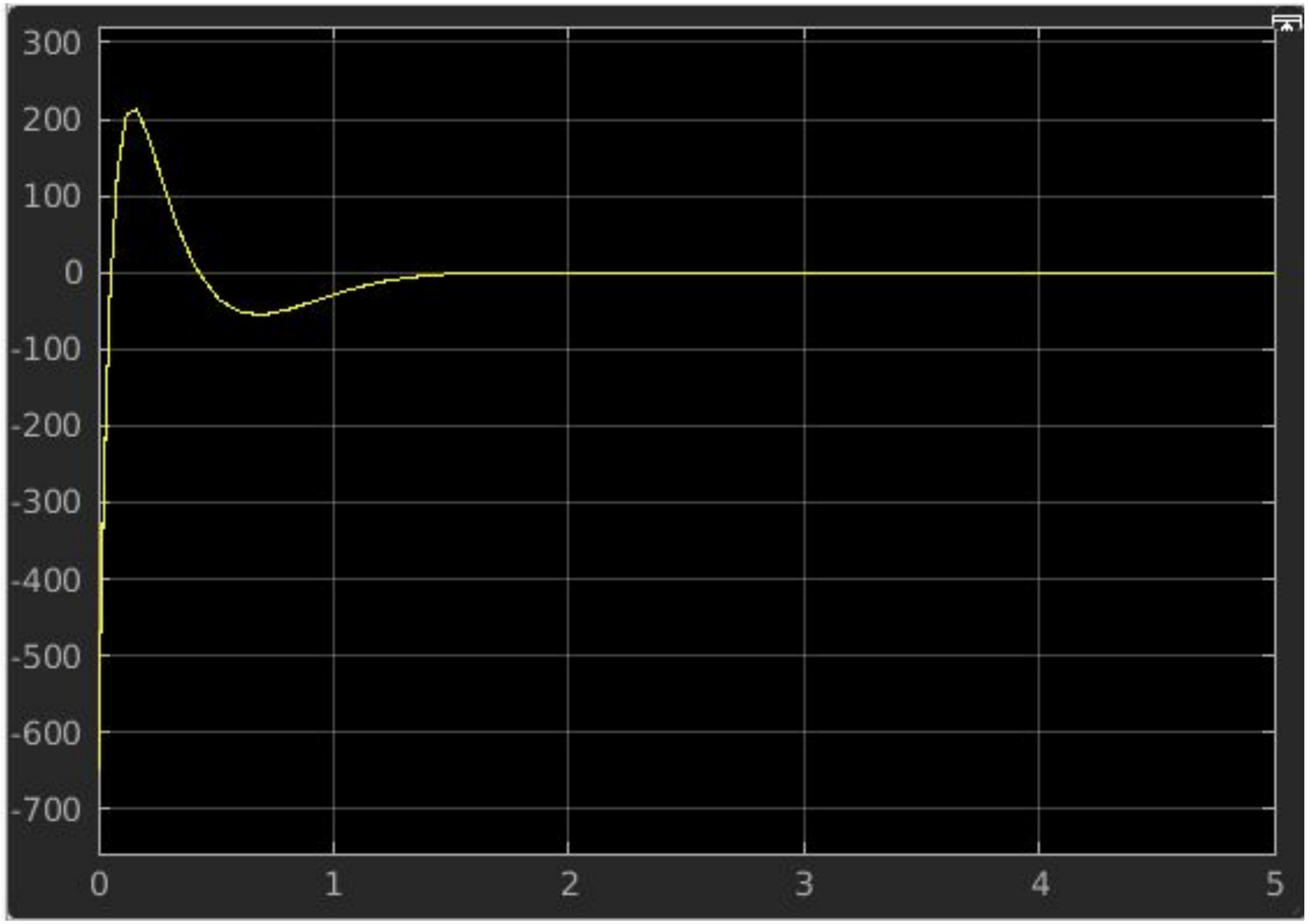


Position vs Time for GA LQR

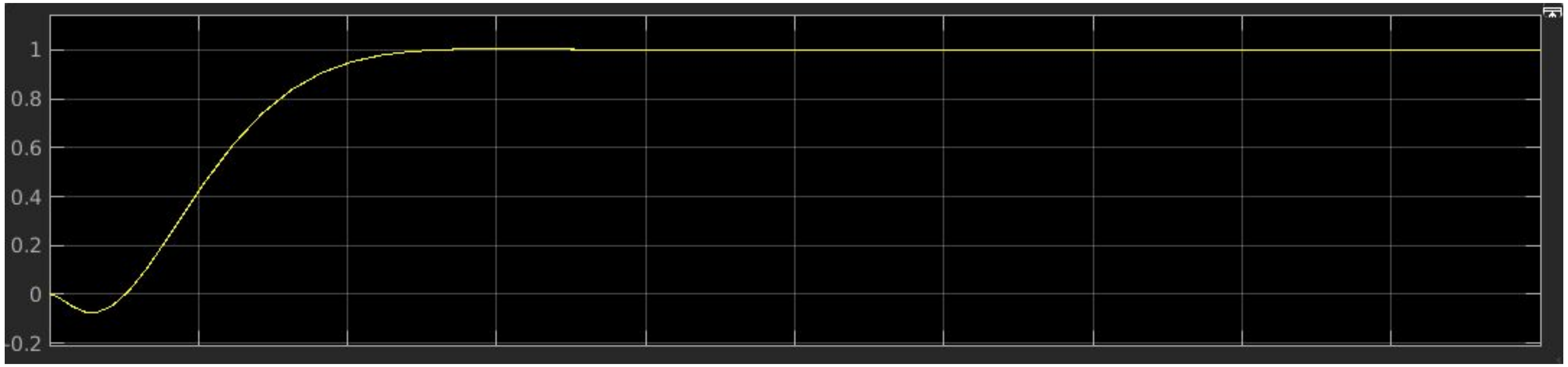


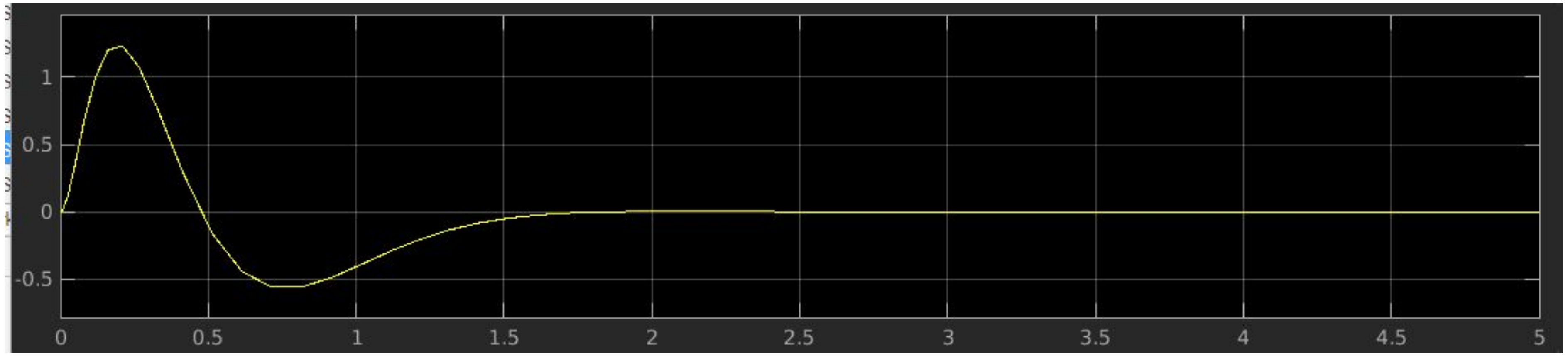
Angle vs Time for GA LQR

GA graph(Voltage and Time)

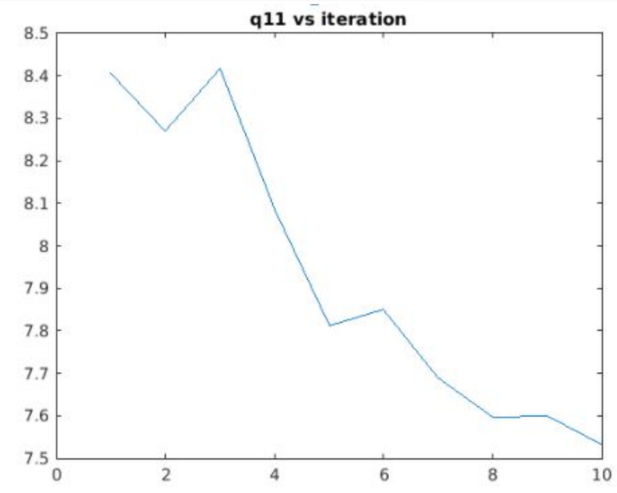


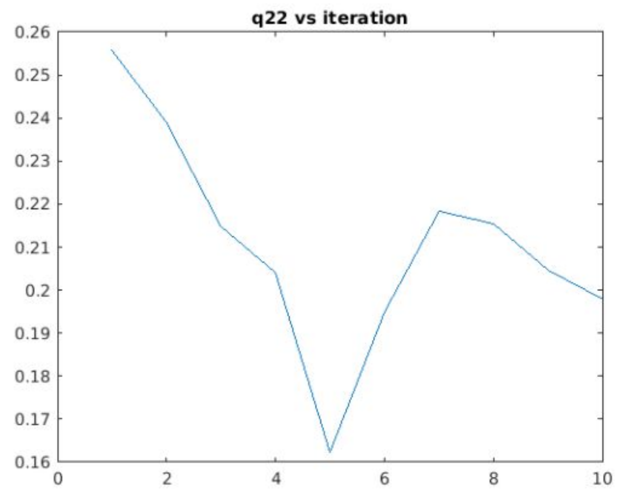
GA graph(position and angle vs time from SimuLink)

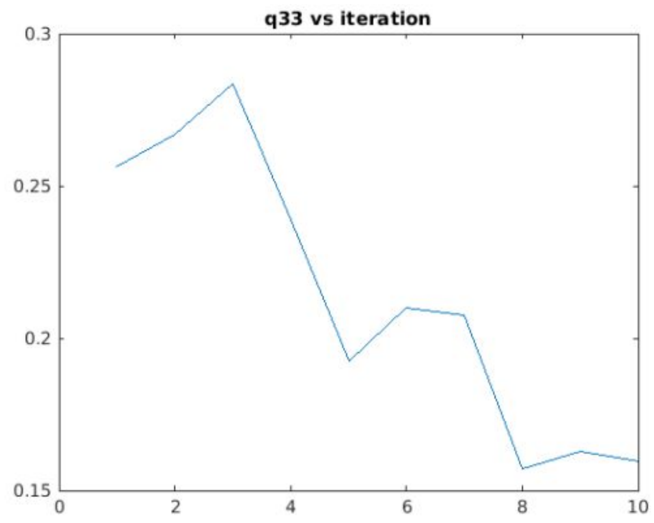


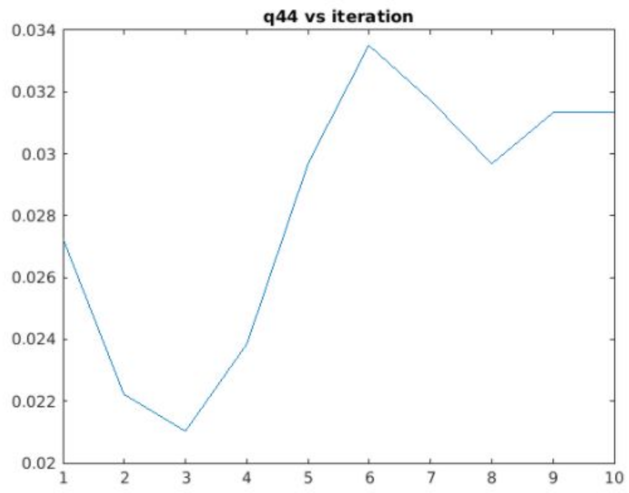


GA graph(q values vs iteration)

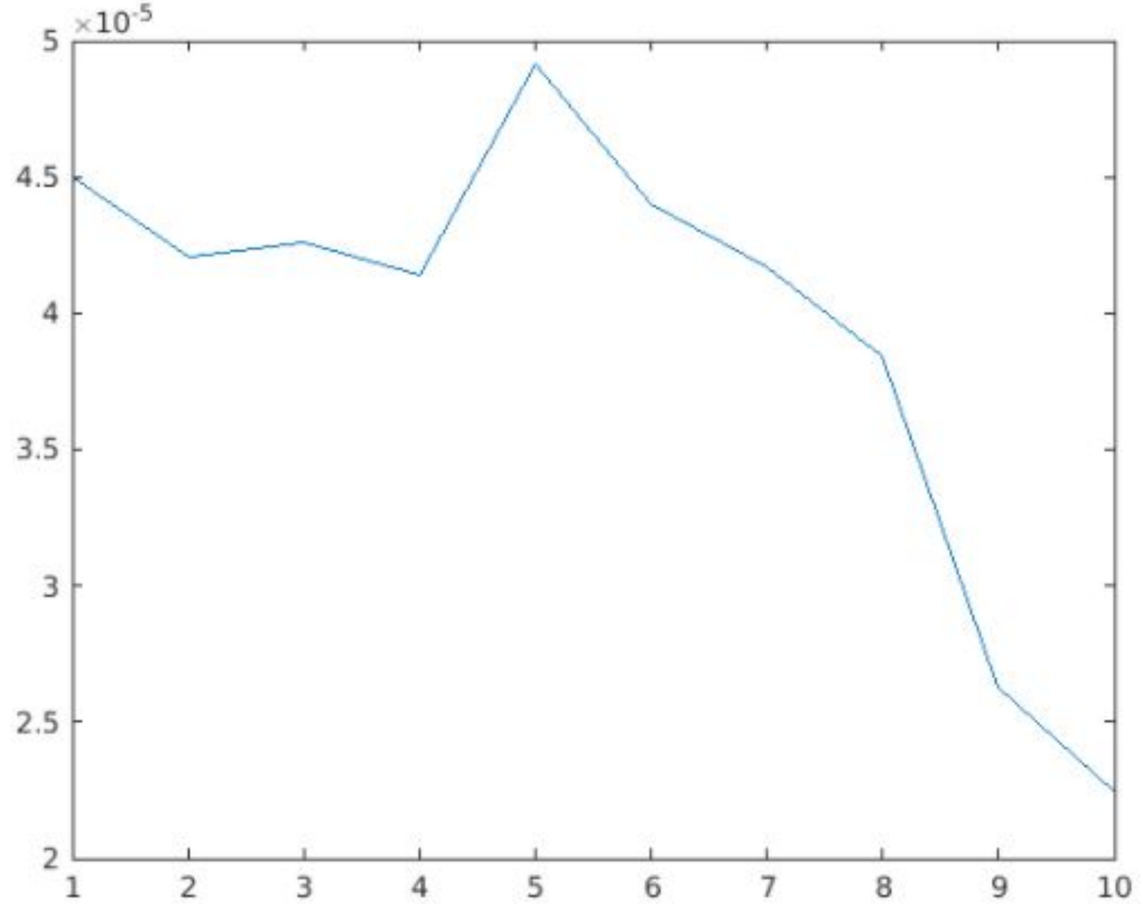








GA graph (r vs iteration)



BFO Parameters

Fitness Function = Rise Time + Settling Time + Overshoot

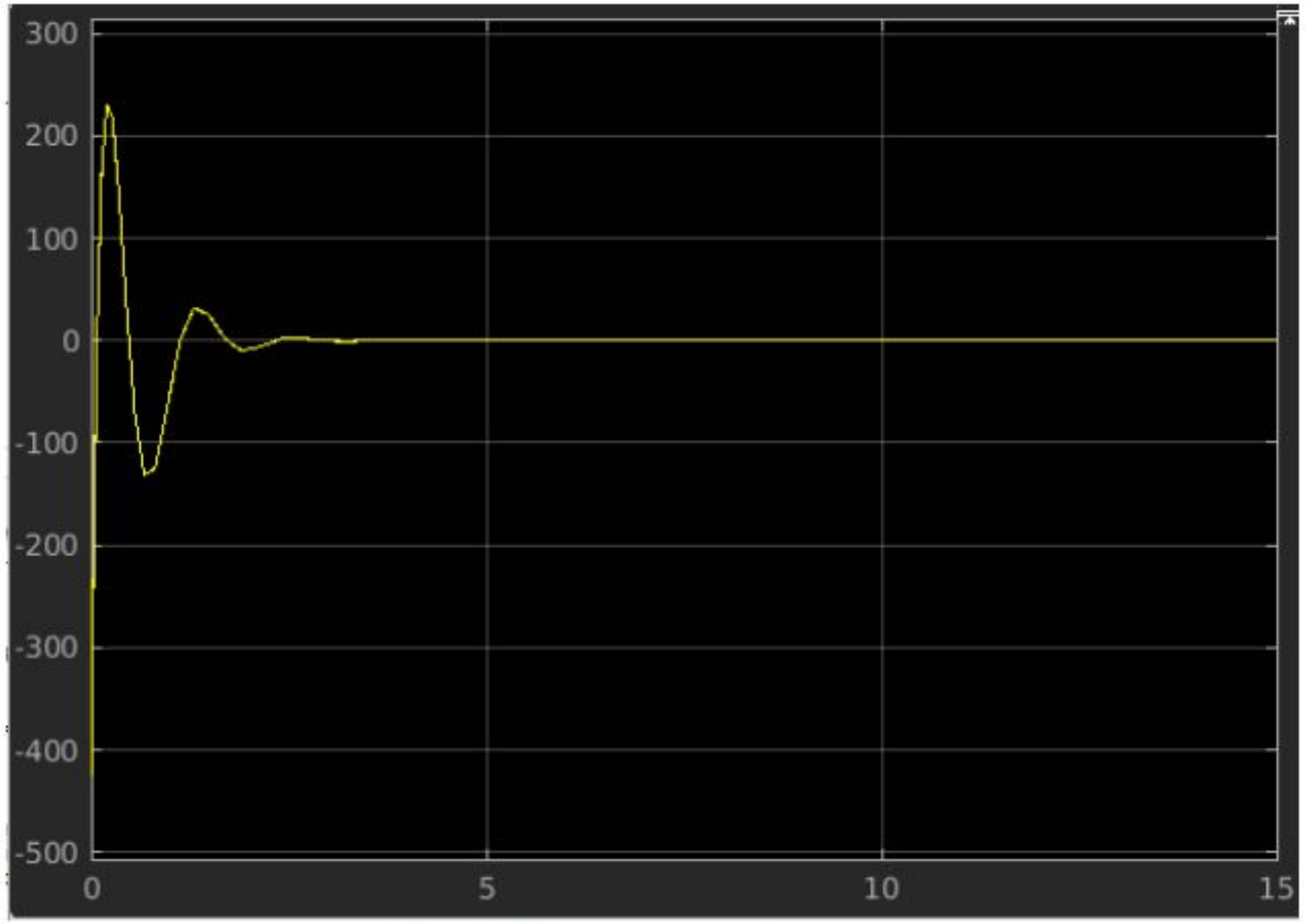
Number of elimination and dispersal steps (Ne)=10;
Reproductive Steps (Nr)=10;
Chemotactic Steps (Nc)=10;
Population Size (Np)=5;
Swim Steps (Ns)=8;
Swarm Size (D)=5;
Probability of elimination (Ped)=0.9;
The run-length unit i.e., the chemotactic step size during each run or tumble (C)=0.01;

Fitness Function Value = 2.170198213995942

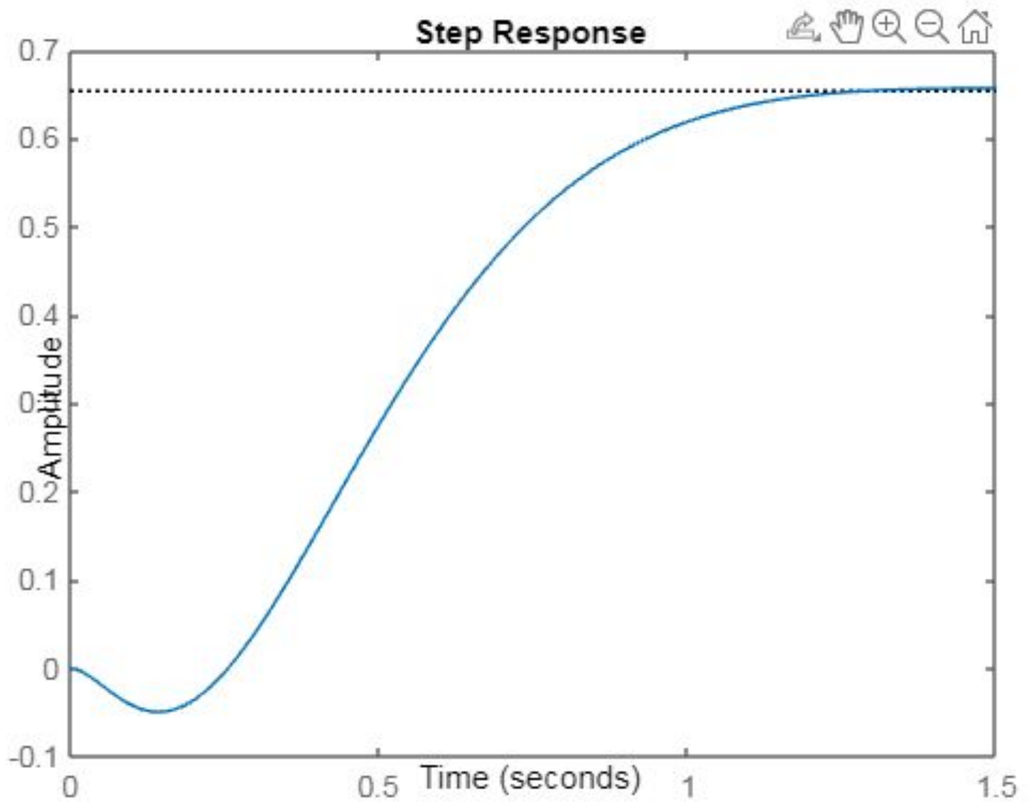
Value of Q and R obtained =
[q11 q22 q33 q44 R] = [q = [72471.87066219141 -0490.898006878771 -04750.45905426314 -05.18027662623499 0.401824162185799];]

Value of K obtained from it =
1.0e+02 * [-4.246847946535302 -2.057666744004326 -2.923004678200982 -0.430108120238977]

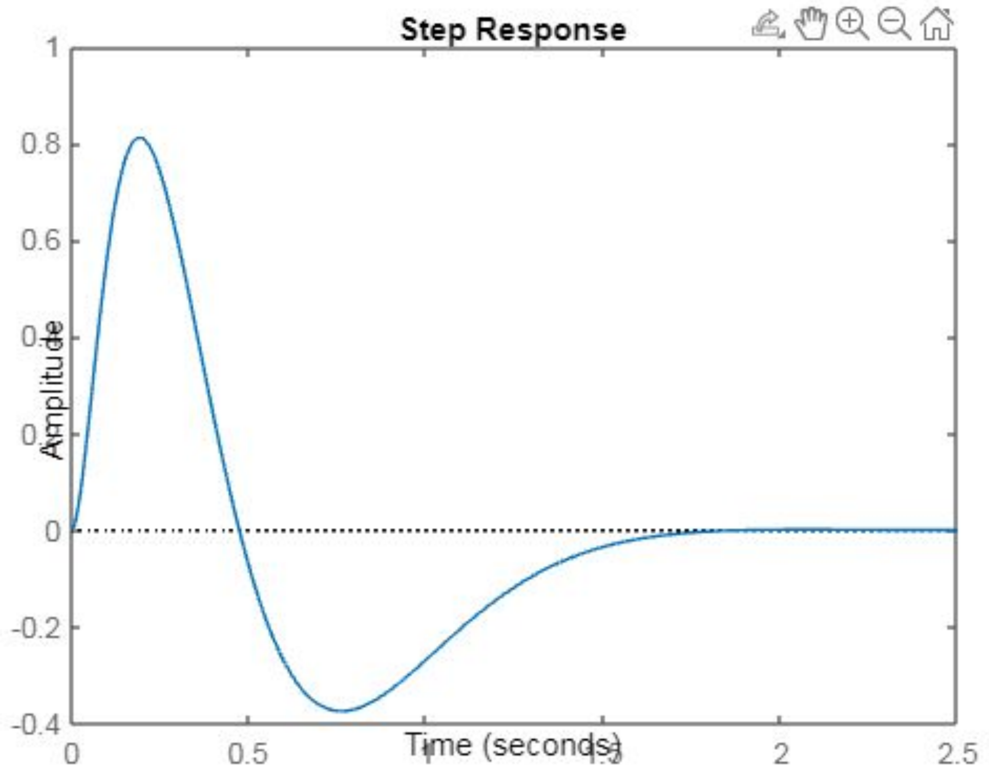
BFO Graphs(voltage vs time)



BFO graph (position vs graph)

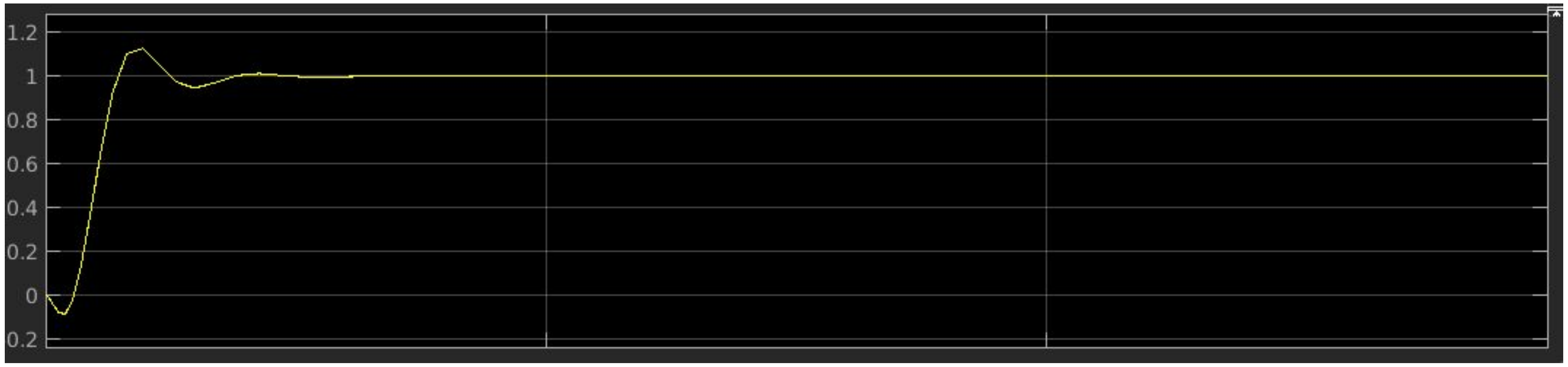


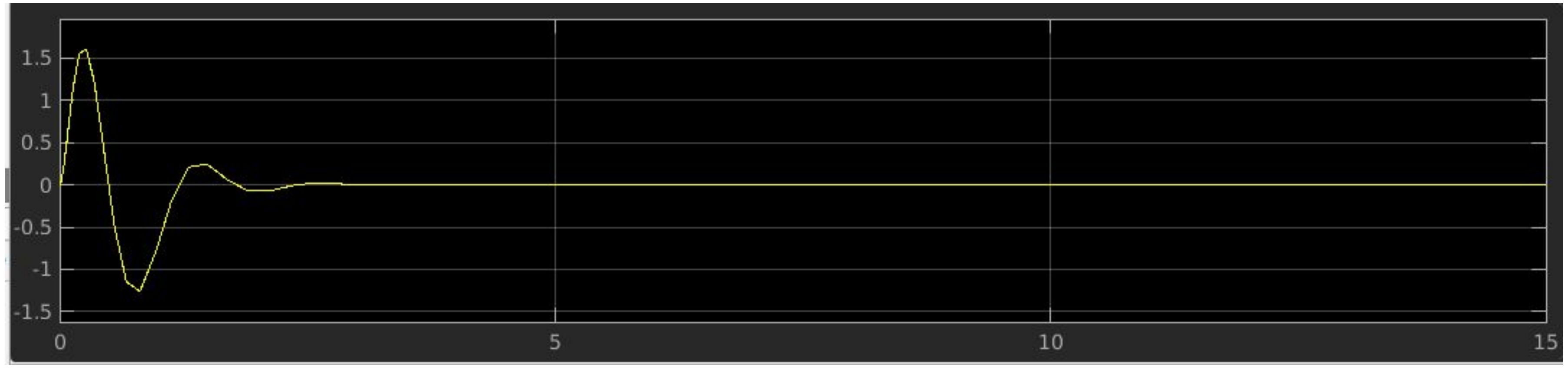
Position vs Time for BFO LQR



Angle vs Time for BFO LQR

BFO graph(Position and angle vs time)





Some points on original results

a) Value of Q,R,K mentioned in the paper for GA optimization

[q11 q22 q33 q44 R] = [8.969 0.308 0.121 0.0085 2.123*10−5]

K = [-6.4988 -3.8592 -5.1982 -0.7348]

But according to the values of Q and R following K should be obtained which provides correct results

K = 1.0e+02 * [-6.499755421319758 -3.859528575379667 -5.198431774768298 -0.734822500368363]

b) Value of Q and R mentioned in paper for BFO algorithm is following

[q11 q22 q33 q44 R] = [289.104 -5.15e-5 -5e-5 -5.1e-5 0.00028]

K = [-1022 -260.2 886.72 77.7]

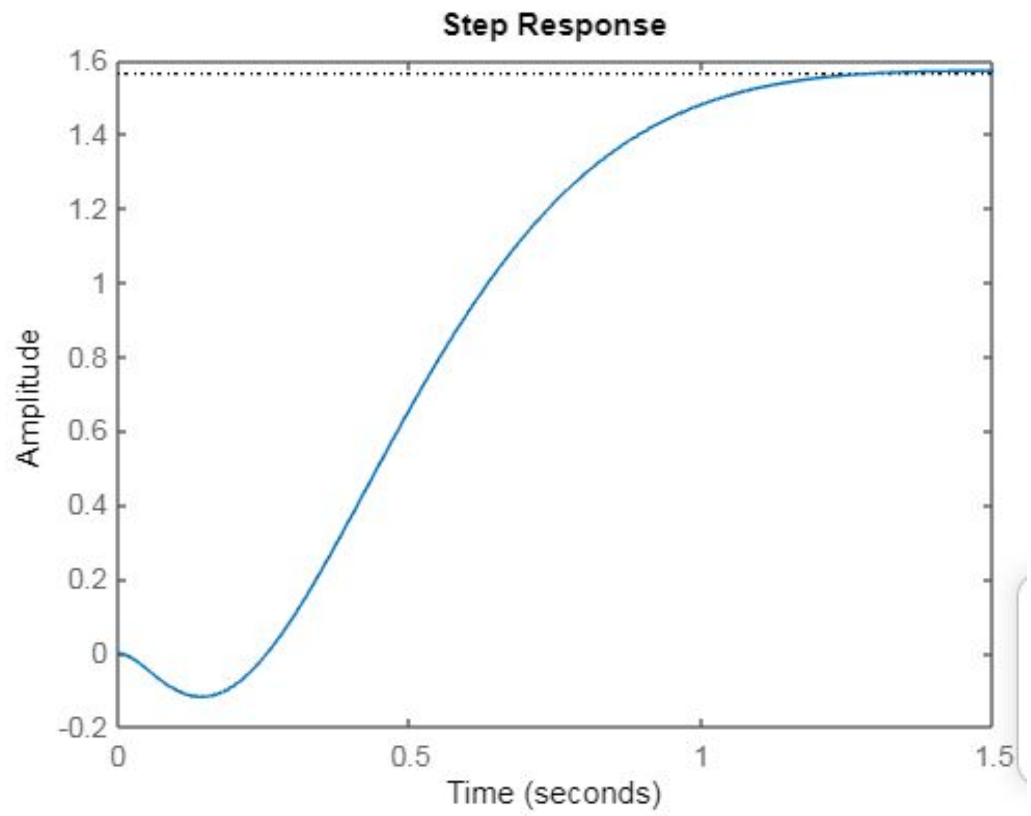
But according to the values mentioned of Q and R value of K should be,

K = 1.0e03 *[-1.016127101161195 -0.462096381004273 -0.527358592608302 -0.072462682693461]

In this magnitude is same but signs are different

And according to the values of K found by us results come this which are wrong according to us. Hence BFO is not providing providing better results according to the values mentioned in the paper

Cont.



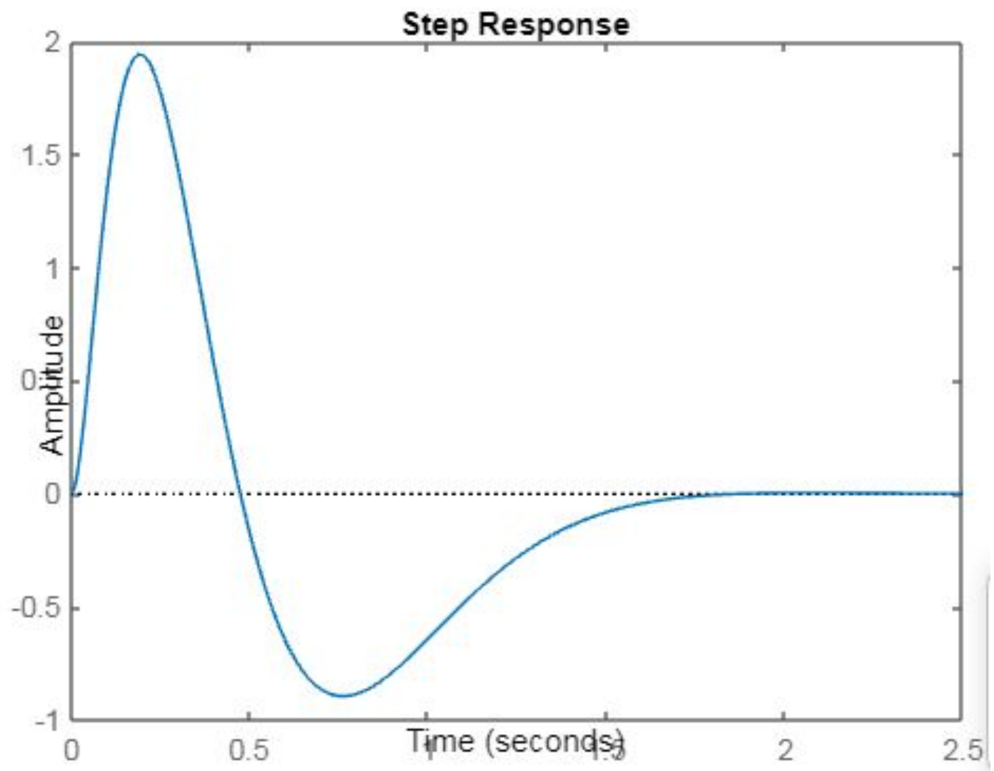


Fig. Variation of Position and Angle according to Q and R values calculated from BFO

Observations and Discussions

A. GA LQR seems to perform better and requires less computational power. It took around 1 sec for execution of GA and 4min for complete execution of BFO. Therefore the conclusion obtained by us is different from that mentioned in paper.

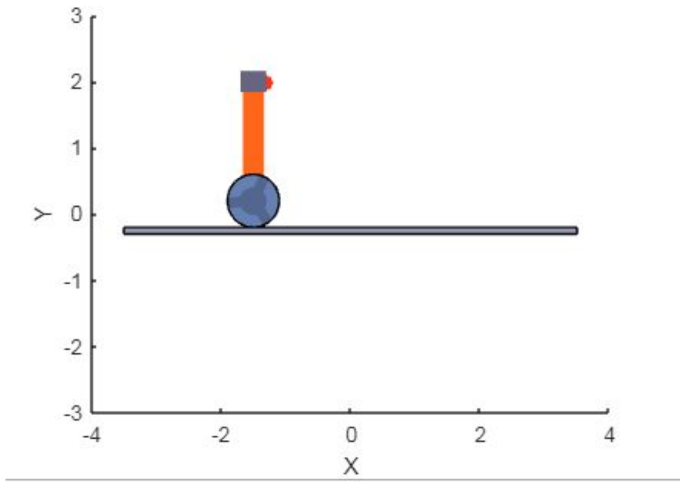
B. Also the value of the fitness function for the most optimized result is the same for GA and BFO whereas that should have been different for both. This is due to the fact that steady state is not an error and is not included in the fitness function.

C. According to our modelling we simulated the results using both Steady state and Transfer function models for a state feedback control scheme. This was different from the standard controller because of the fact that modelling a TF model for such a controller needs heavy mathematical manipulations.

Learnings

A. For our model, we incorporated a deviational set-point scheme (not present in paper) and provided it with a step input to see the results. The graphs were obtained as expected after mathematical manipulations and tunings.

B. We also tried using an animation model, made by Columbia University professors, into our model to see the results of the segway model. But, we could not get the running animation, but only the final figure :



Conclusions.

- In this work, an optimal linear control system was adopted to balance a Segway two-wheeled mobile robot.
- First state space model of the plant was derived and the model was simulated both in MATLAB and SIMULINK for a step input to position.
- To improve the results for LQR two biologically inspired algorithms were implemented, GA and BFO, to find optimal value of Q and R.
- According to the obtained results it was found that GA LQR had better performance than BFO. Though the mentioned conclusion is different from one that is mentioned in the paper but values provided by them also provides simulation results similar to that done by us.

Future Scope

- The developed model is a baseline for many other research works. The use of a simple inverted pendulum controller scheme has been extended to our model. We have not incorporated the turning scenario which would involve non-uniform voltage inputs to both the wheels.

- We have only used a state-feedback LQR for tuning the controller gains. Instead many other controllers like PID and its variants(like Fuzzy and FOPID) could be used. Even a simple Q-learning based approach could also be incorporated.

- Similarly, for tuning the controller parameters, we could have used any other data-based(DL) and data-free methods. We had planned to use PSO and ACO along with BFO and GA, but were not able to complete it on time.

- Further, the distance-speed graphs obtained can be sampled and provided to any navigation algorithm using Artificial Potential field and be used for autonomous navigation of the segway. This hardware model can be easily applied to any microcontroller.

- State estimation Kalman filters could have been used to find the accurate values of the output variables, and this could have made the model more robust and real-world. In our output graphs, we do not see ripples, which would have been manifested by using state estimation filter.

innovate achieve lead

Thank You