Reinforcement Learning based Non-Linear Flight Control

Subrahmanya V Bhide (SC18B030)

under the guidance of Dr. Dhayalan R

Indian Institute of Space Science and Technology

May 26, 2022

Motivation

- Current aircraft autopilots use PID controllers.
- PID controllers are based on the linearized approximations of the governing equations.
- It has been shown that RL can actually perform better than PID controllers for quadcopters.
- We apply RL for conventional aircrafts.

Reinforcement Learning

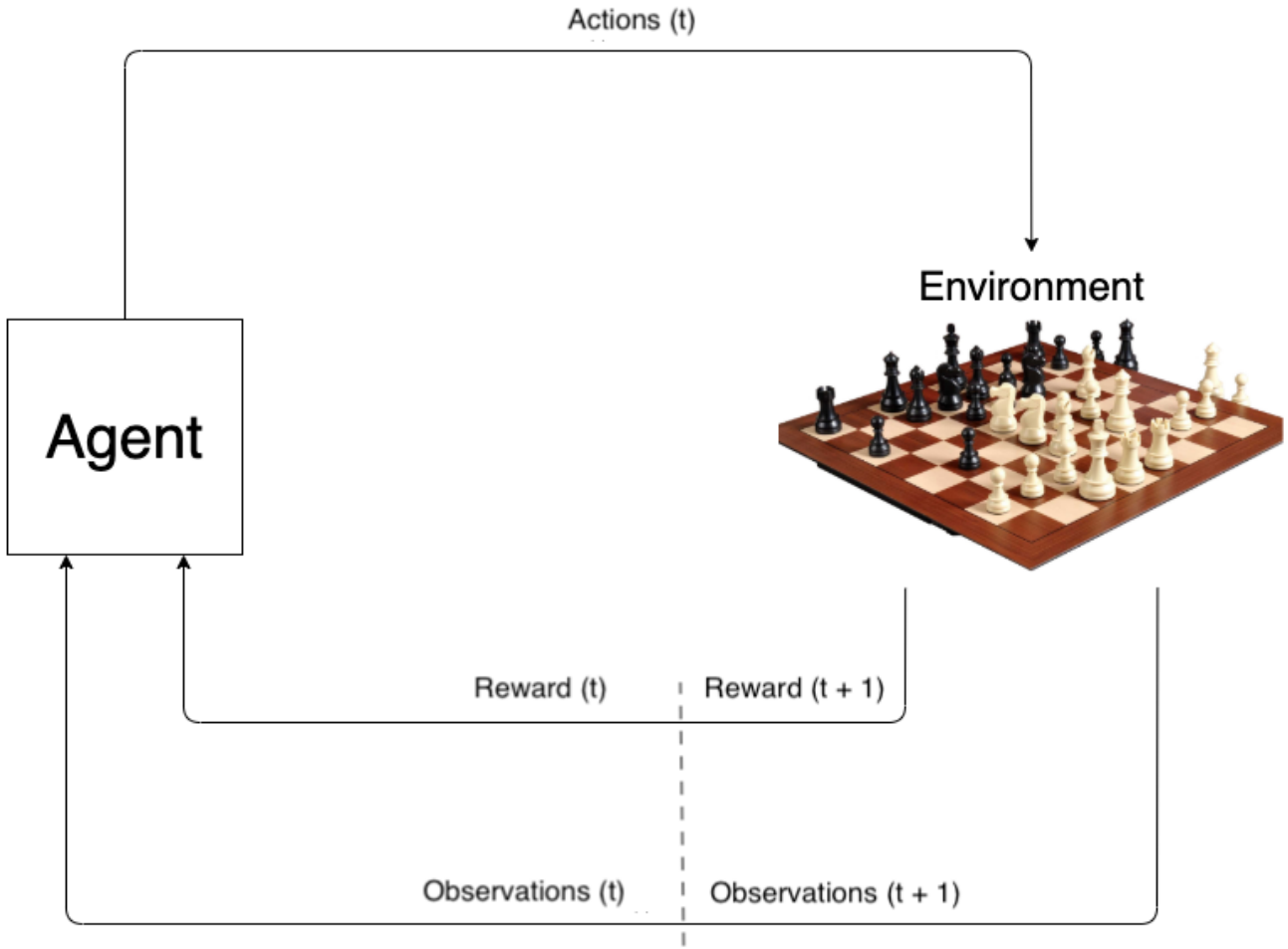
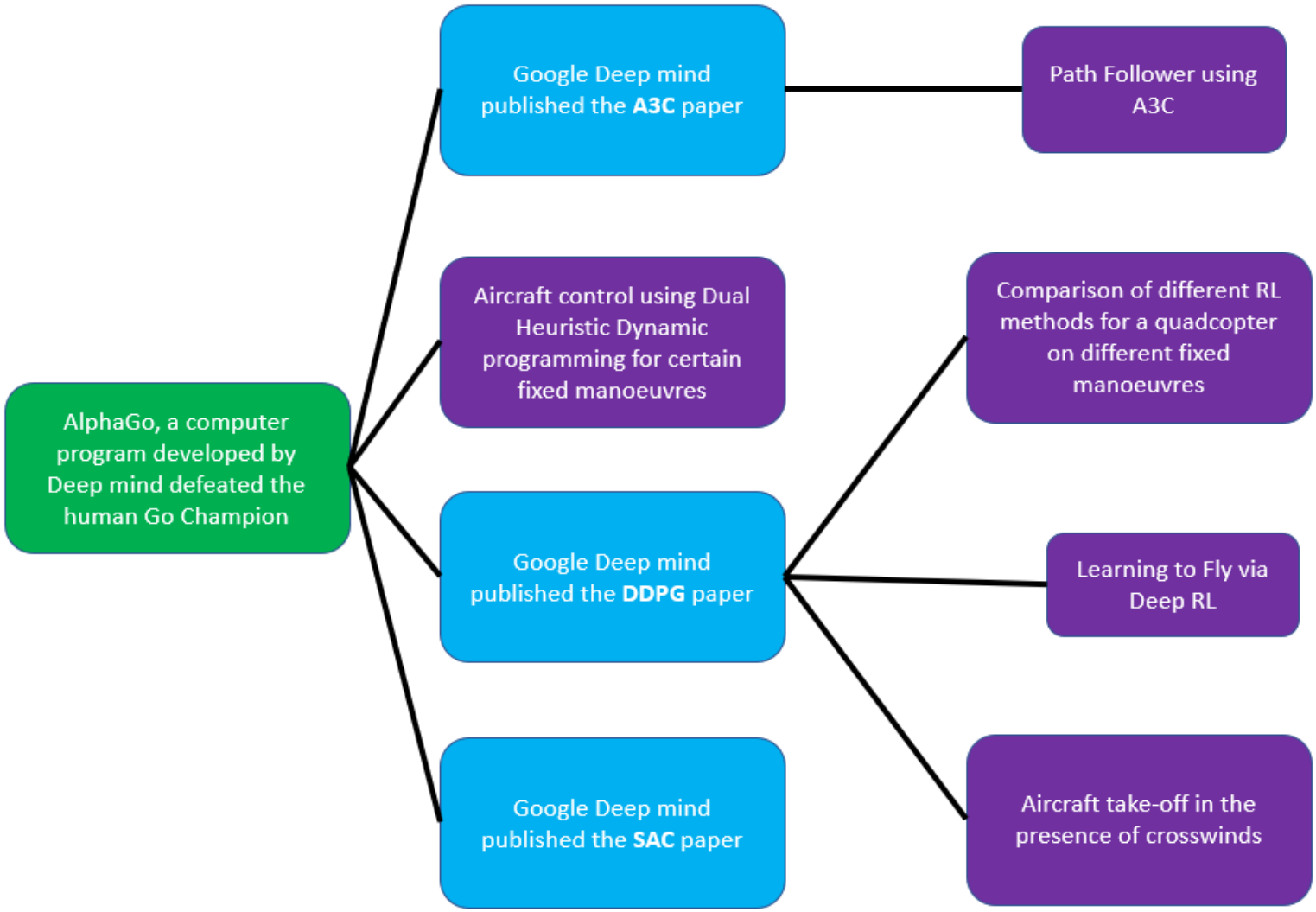


Figure: A general RL model

Existing Literature



Setting the objectives

- Developing a controller for a conventional aircraft, such that given a state X and a goal state X_{goal} the controller provides the control inputs $\delta = \{\delta_e, \delta_a, \delta_r, \delta T\}$.
- Once such a controller is achieved, by varying the X_{goal} state appropriately with time, different maneuveres can be achieved.
- And by keeping the X_{goal} constant, steady flight can be achieved.
- The state vector for the aircraft can be suitable measurable parameters and all the X_i together must be able to influence the control inputs.

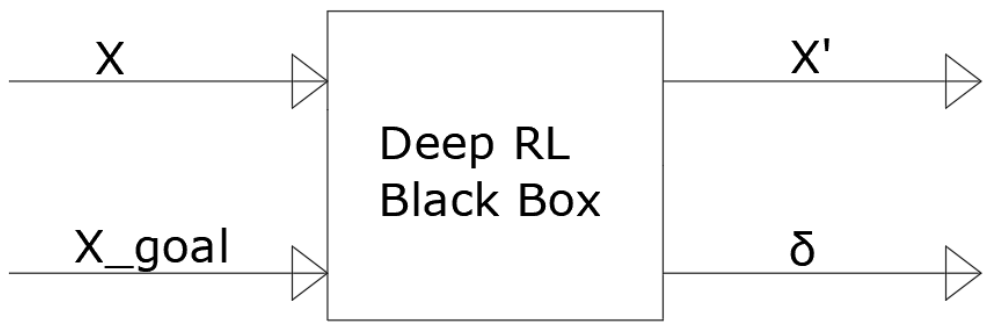


Figure: Flowchart for the controller to be developed.

Approaching the problem

1. Why reinforcement learning ?

2. Longitudinal and Lateral-Directional dynamics.
   In the longitudinal dynamics the states would be
   $X = \{u, w, q, \theta\}$; $\delta = \{\delta_e, \delta T\}$.

3. Pure pitching dynamics.
   $X = \{q, \theta\}$; $\delta = \{\delta_e\}$

Equations of Motion

The equations of motion for the pure pitching motion are:

$$\dot{q} = \frac{M}{I_{yy}}$$

$$\dot{\theta} = q$$

- Here $M = q_\infty S \bar{c} C_m$ and $C_m = C_{m_0} + C_{m_\alpha}\alpha + C_{m_{\delta_e}}\delta_e$

RL algorithims implemented I

- Deep Determinsitic Policy Gradient

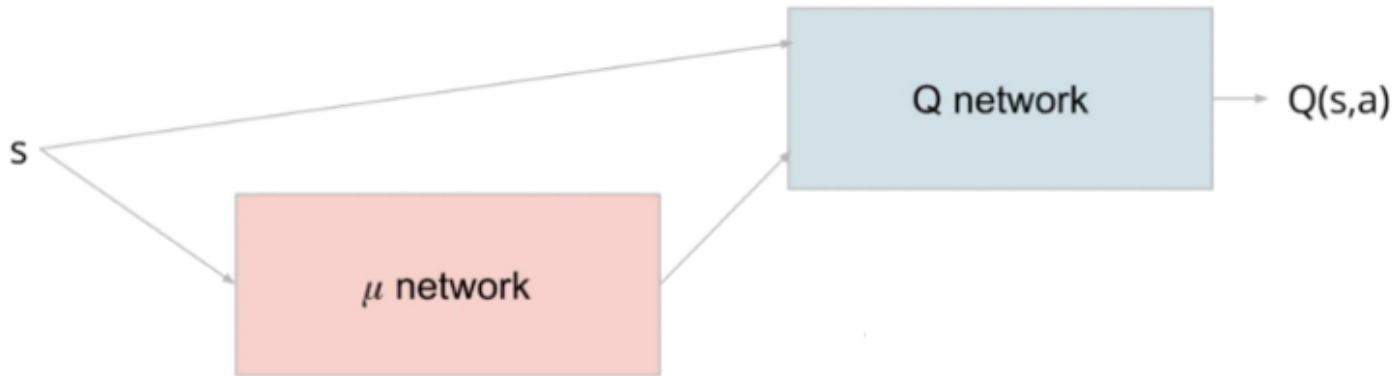


Figure: A representation of a DDPG agent.

- Soft Actor Critic

$$J_\pi = E_\pi \left[\sum_t r(s_t, a_t) - \alpha \cdot log(\pi(a_t|s_t)) \right]$$

RL algorithims implemented II

- Asynchronous Advantage Actor Critic Method

RL algorithims implemented III

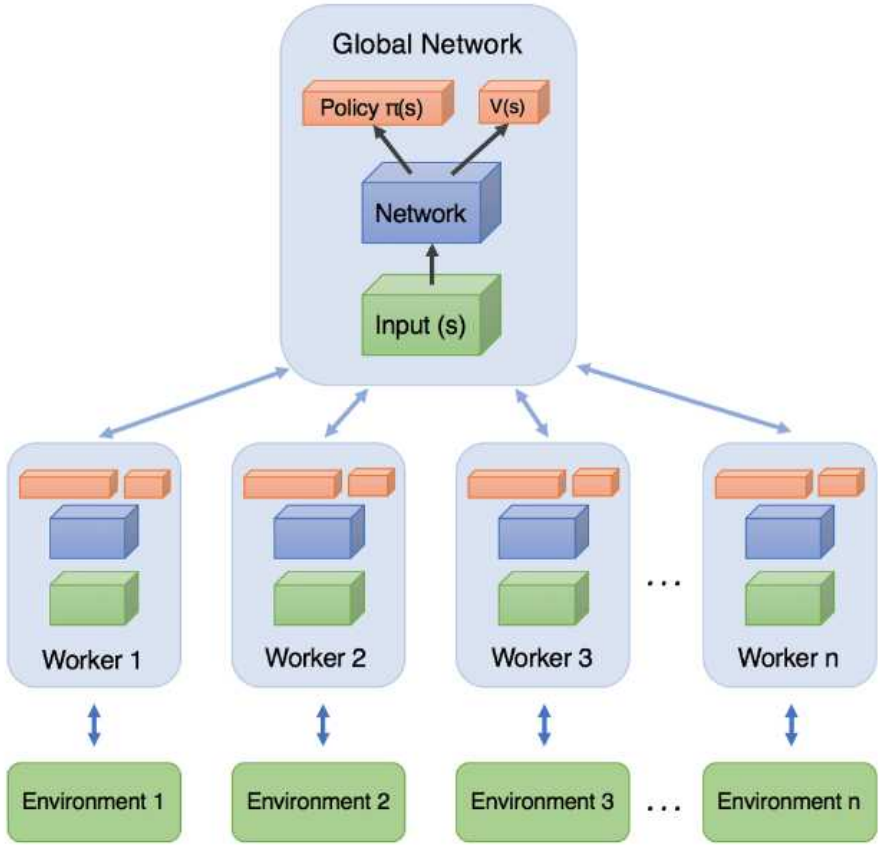


Figure: A representation of a A3C agent.

RL Training Methodology

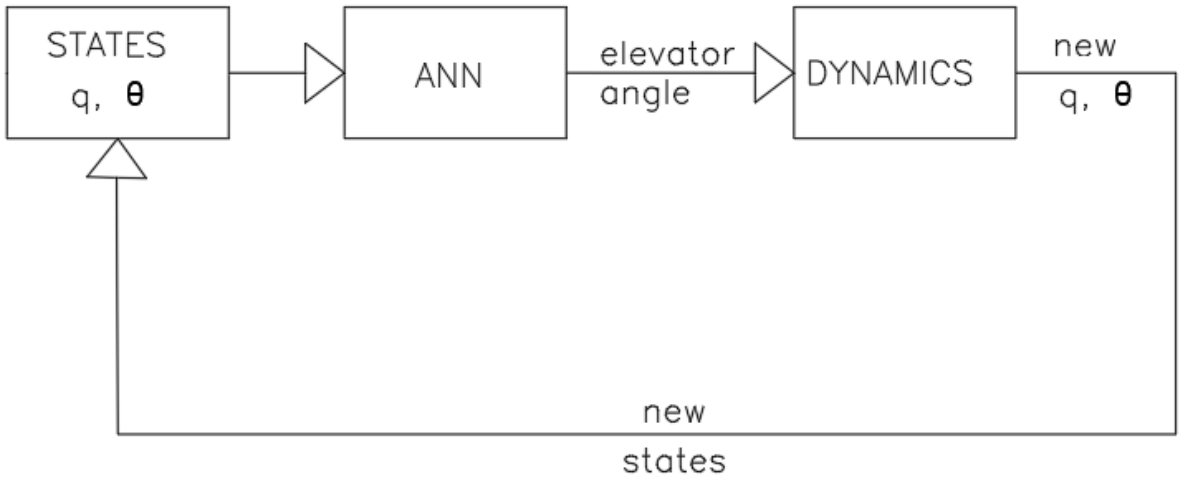


Figure: Flow diagram showing the sequence of operations for the controller to be developed

- How to include X_{goal} information ?

State Vector

The following state vectors were used for the analysis,

1. $X = \{q, \Delta\theta, \theta_{goal}\}$
2. $X = \{q, \Delta\theta, \dot{q}\}$
3. $X = \{q, \Delta\theta, \theta\}$

$X = \{q, \Delta\theta, \theta\}$ gave the best performance of the three state vectors

The initializations for the different elements in the state vector are done as follows:

1. $\Delta\theta$ is initialized randomly between -5° and 5°.
2. q is initialized randomly between -0.01 rad/s and 0.01 rad/s, doing so provides robustness to the algorithm.
3. For cases where $\dot{q}$ is used as a state, it is initialized to zero.
4. The episode is terminated in cases where, $|\theta|$ goes beyond 10° and the elevator can vary between -15° and 15°.

Neural Network Paramters

Initially training for different RL agents with ANN's with 1, 2, 3 and 4 hidden layers and 8, 16, and 32 neurons was done.

- Net 1, Actor and Critic

  Layer 1 - 32 neurons - ReLu

  Layer 2 - 32 neurons - ReLu

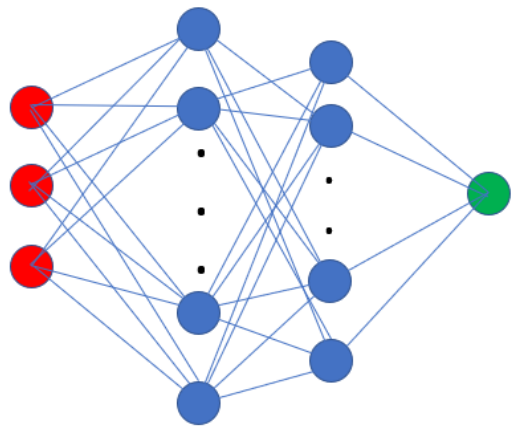
  Output- 1 neuron - Linear

- Net 2, Actor/Critic

  Layer 1 - 400/300 neurons - ReLu

  Layer 2 - 400/300 neurons - ReLu

  Output- 1 neuron - Linear



The values of different hyperparameters that are experimented with are given below:

1. Learning rate : 1E-3, 3E-3, 1E-4. The best results are obtained by using 1E-4.
2. Discount factor : 0.4, 0.44, 0.5, 0.8, 0.9, 0.99. The best result was obtained by using $\gamma = 0.99$.
3. Timestep : 0.02s. This results in a update frequnecy of 50Hz.

Reward Functions I

Reward function engineering is an important way to convery information to the agent apart from the environment. This is a more nuanced method. The many reward functions tried are given below:

- Quadratic variation

$$R = -(x - x_{goal})^2$$

Reward Functions II

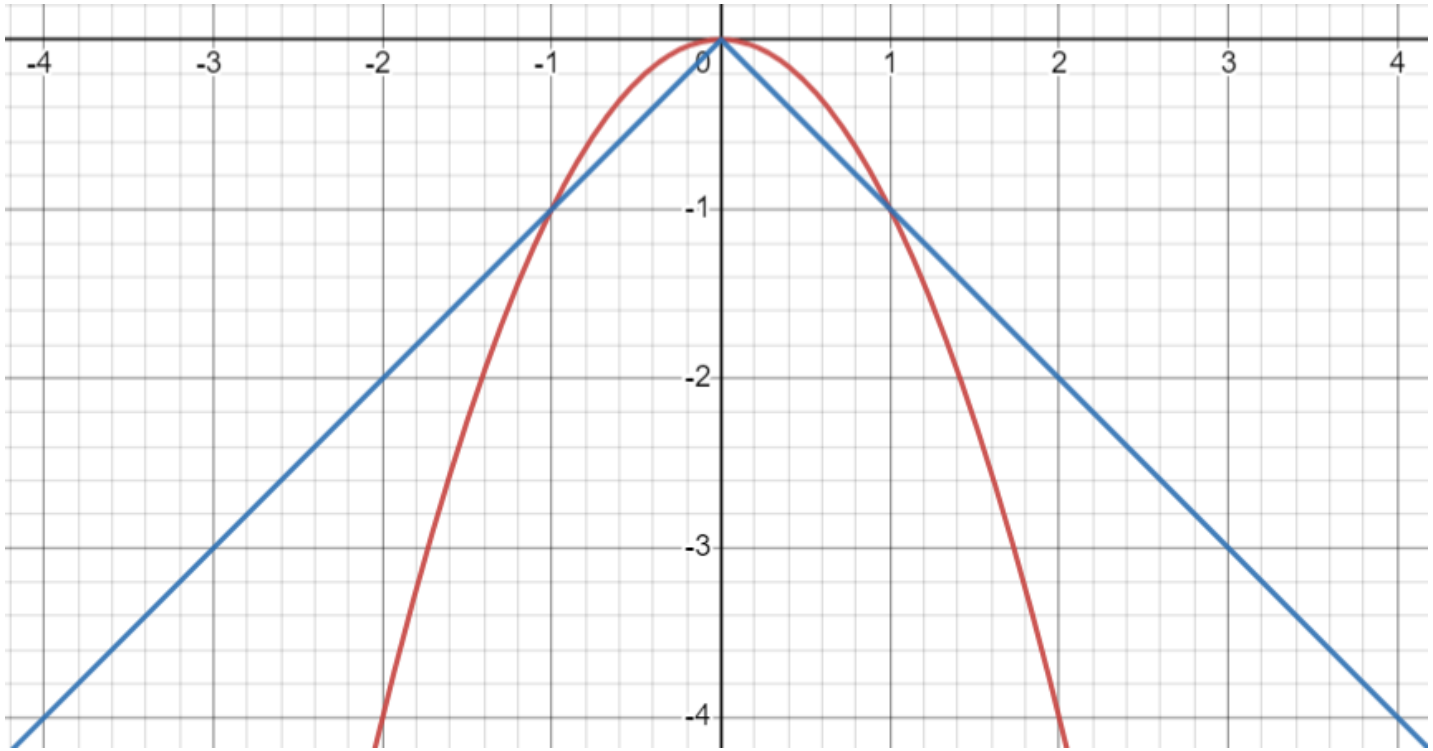


Figure: Quadratic and Linear variations of Rewards

- Quadratic variation conditional reward

Reward Functions III

$$R = dx_{set}^2 - (x - x_{goal})^2$$

when $dx_{set} > |x - x_{goal}|$

- Ideas from PID controllers

$$R = -\frac{1}{2}(e_x + e_{x-1}) \cdot dt$$

$$R = -|PID - \delta_e|$$

- Gaussian, quite opposite to the quadratic variation

$$R = exp(-e_x^2)$$

- Elevator Reward function

$$R = -|\Delta\delta_e|$$

- Scaling up of Reward function terms

	q	$\Delta\theta$	$\dot{q}$
Typical Value	0.01	0.017	6.25
Scaling factor	10000	10000	1

Results I

1. Initial DDPG trial Net 1,

$$R = -20\Delta\theta^2 - q^2$$

and for $|\Delta\theta| < 1^\circ$

$$R = R + 100 \cdot (1^{\circ^2} - \Delta\theta^2)$$

2. The possible reasons for the non-convergence of the DDPG algorithm in the case are:
   - 2.1 More training is required for the DDPG to converge and provide an optimal solution.
   - 2.2 The DDPG algorithm somehow doesn't suit this problem.
   - 2.3 The input state required to make the decision are not good enough and some more information is needed.
3. For the same reward function an agent is trained using A3C algorithim with Net 1.

Results II

4. Soft Actor Critic, Net 1

$$R = -20 \cdot \Delta\theta^2 - q^2 - |\Delta\delta_e|$$

If $|\Delta\theta| < 1^\circ$,

$$R = R + 100 \cdot (1^{\circ^2} - \Delta\theta^2)$$

5. Change of states to $\{q, \Delta\theta, \dot{q}\}$ and weighing the reward function terms,

$$R = -10000 \cdot \Delta\theta^2 - 10000 \cdot q^2 - \dot{q}^2$$

If $|\Delta\theta| < 1^\circ$,

$$R = R + 10 \cdot (1^{\circ^2} - \Delta\theta^2)$$

Results III

6. Rewards such as PID based reward functions, bonus and penalty based quadratic reward functions, gaussian functions are used for training.

$$R = -\Delta\theta^2 - q^2 - |\Delta\delta_e|$$

If $|\Delta\theta| < 1^\circ$,

$$R = R + 10 \cdot (1^{\circ^2} - \Delta\theta^2)$$

If $|\Delta\theta| > 3^\circ$,

$$R = R - 10 \cdot \Delta\theta^2$$

$$R = -|0.9059 \cdot e_\theta + 0.2738 \cdot \int e_\theta + 0.4407 \cdot \frac{de_\theta}{dt} - \delta_e|$$

Results IV

7. Revisting DDPG using Net 2 and the state vector being $(q, \Delta\theta, \theta)$ and the reward function being

$$R = -\Delta\theta^2 - q^2 - |\Delta\delta_e|$$

but in this case, θ is taken in degrees.

8. Bonus included in the quadratic function

$$R = -\Delta\theta^2 - q^2 - |\Delta\delta_e|$$

If $|\Delta\theta| < 1^\circ$,

$$R = R + 10 \cdot (1^{\circ^2} - \Delta\theta^2)$$

The results obtained for the DDPG agent using Net 2 and the above reward function and a $\gamma = 0.99$ are shown in the following slides

Results V

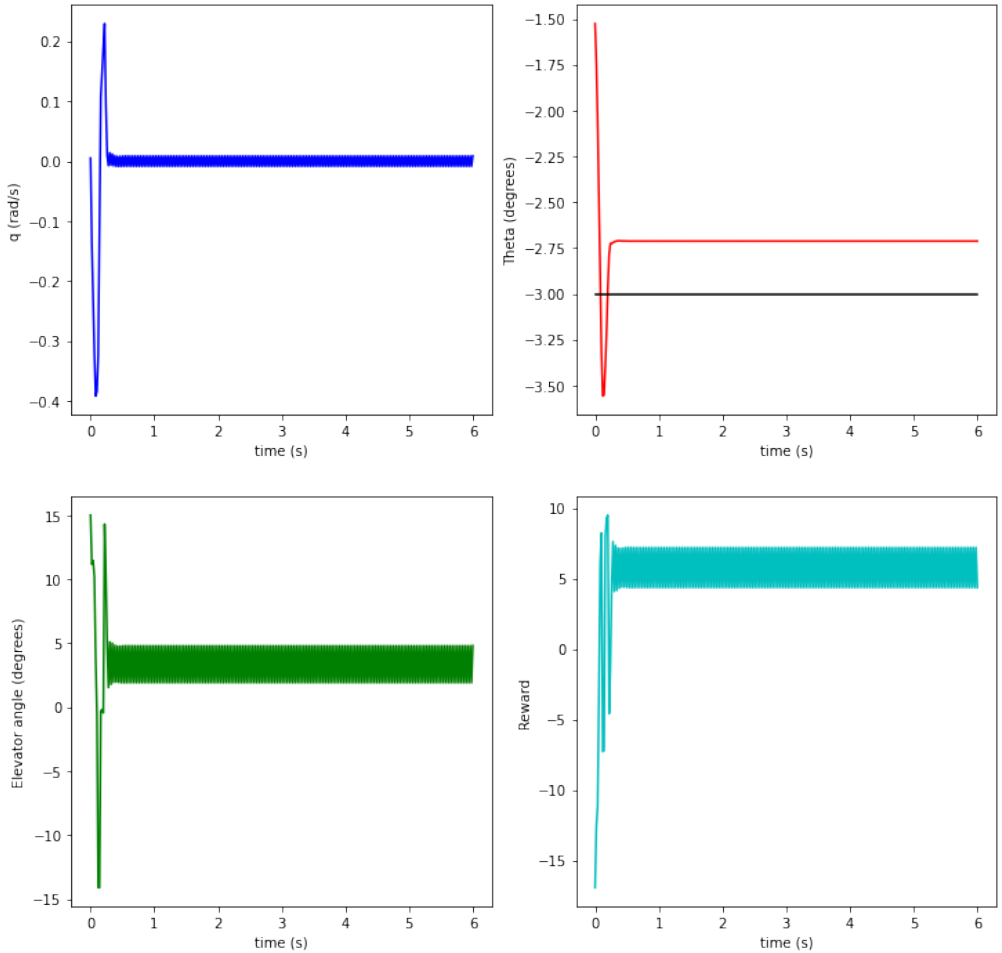


Figure: DDPG agent trained using the state $(q, \Delta\theta, \theta)$ using the second training methodology for a $\theta_{goal} = -3°$

Results VI

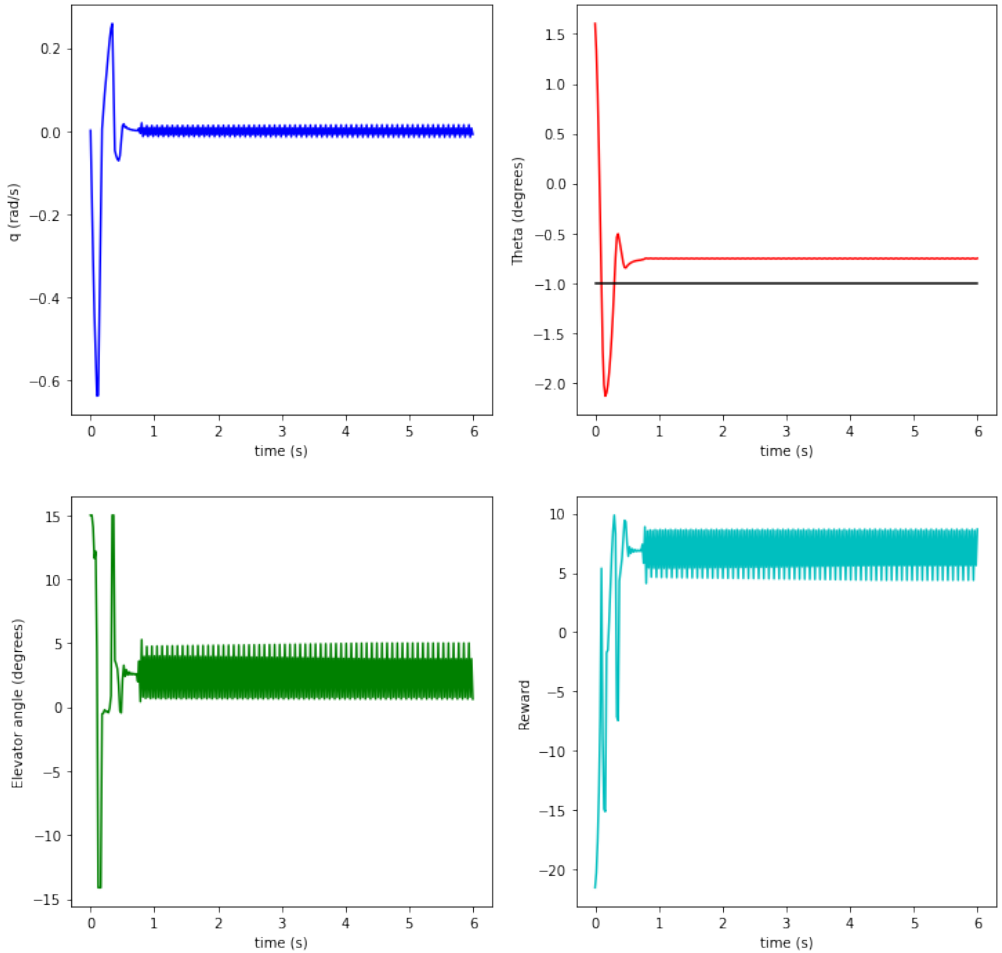


Figure: Results obtained for DDPG agent trained using the state $(q, \Delta\theta, \theta)$ using the second training methodology for a $\theta_{goal} = -1°$

Results VII

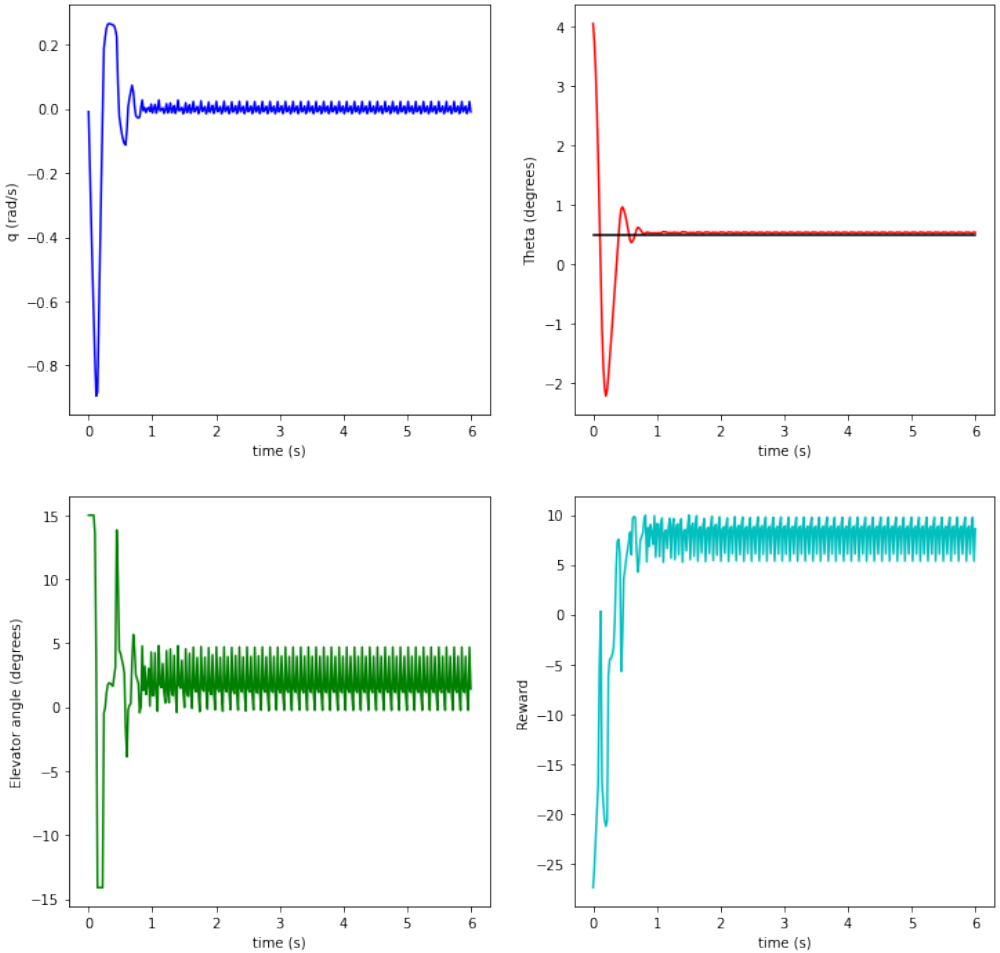


Figure: Results obtained for DDPG agent trained using the state $(q, \Delta\theta, \theta)$ using the second training methodology for a $\theta_{goal} = 0.5°$

Results VIII

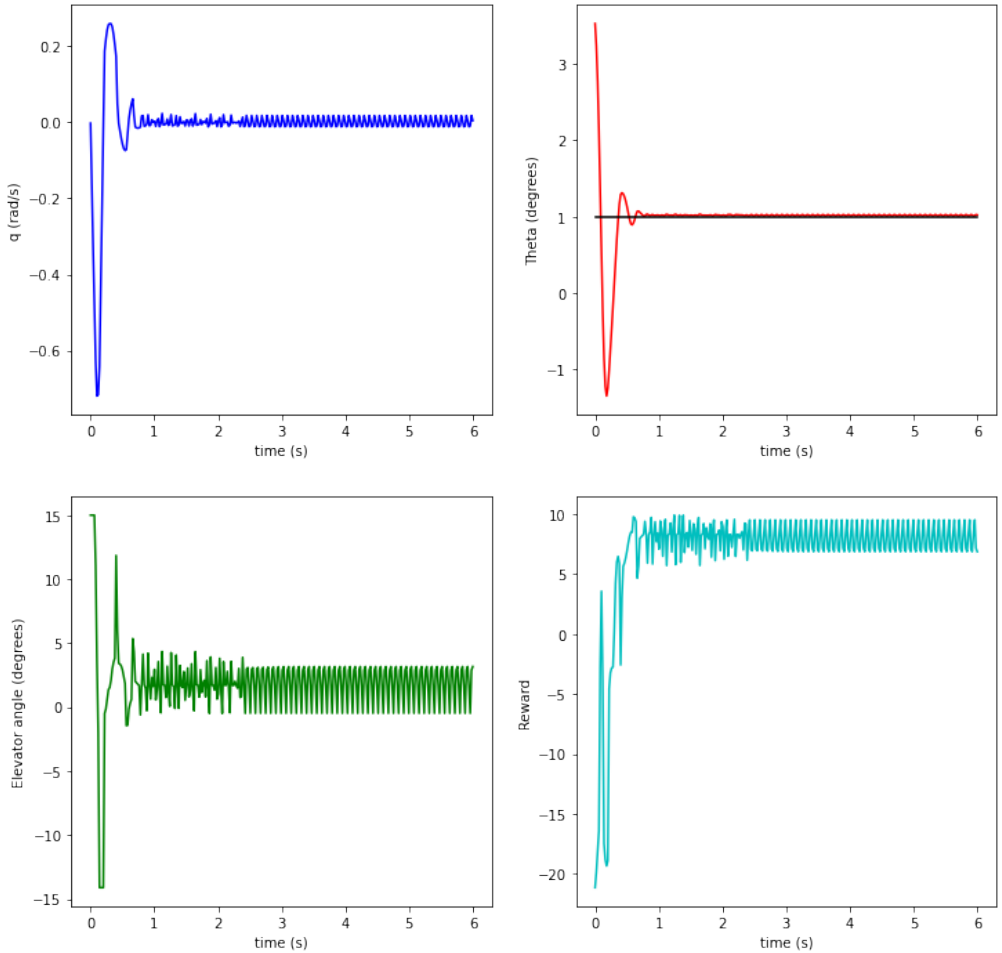


Figure: Results obtained for DDPG agent trained using the state $(q, \Delta\theta, \theta)$ using the second training methodology for a $\theta_{goal} = 1°$

Results IX

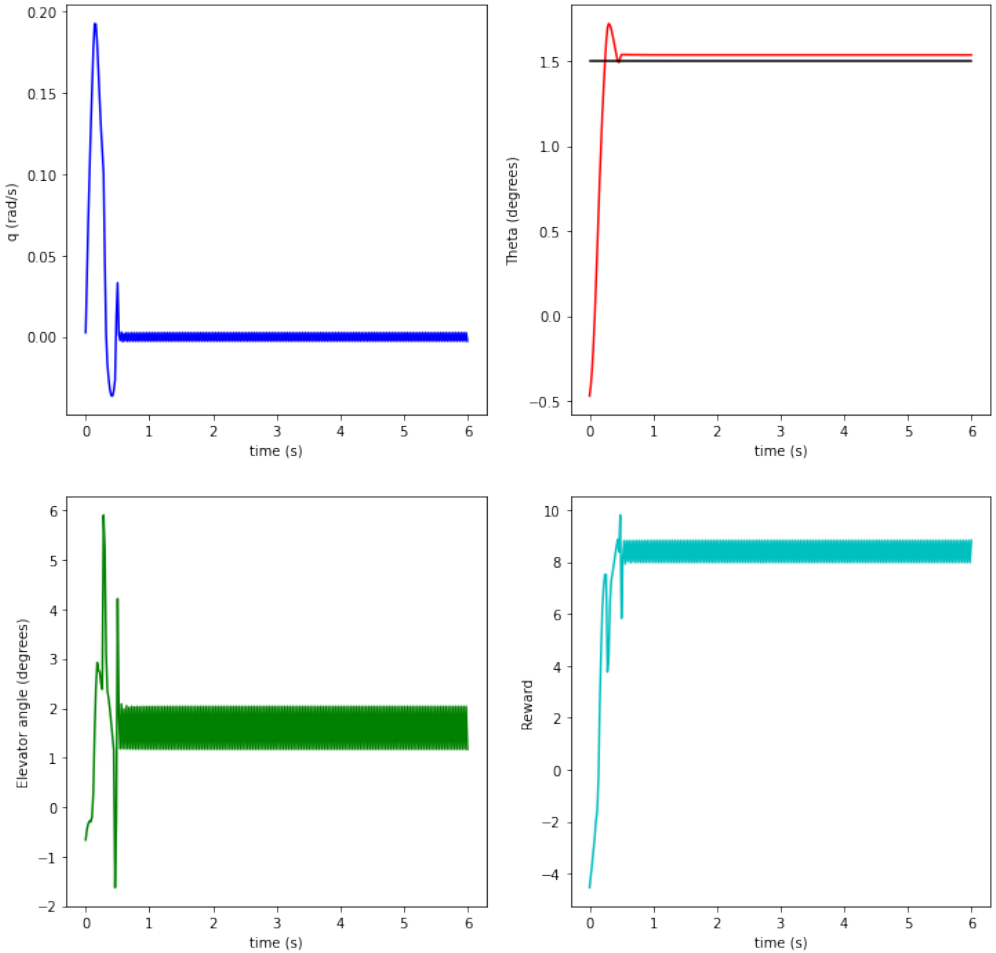


Figure: Results obtained for DDPG agent trained using the state $(q, \Delta\theta, \theta)$ using the second training methodology for a $\theta_{goal} = 1.5°$

Results X

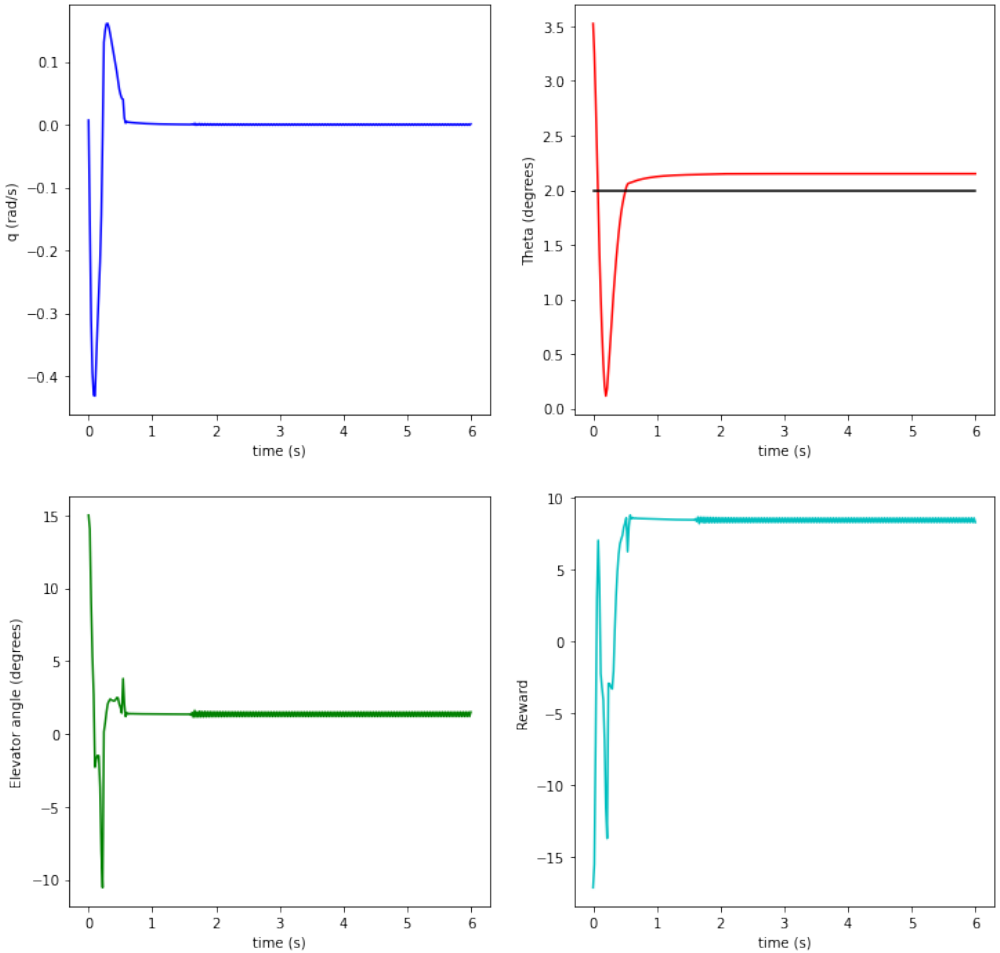


Figure: Results obtained for DDPG agent trained using the state $(q, \Delta\theta, \theta)$ using the second training methodology for a $\theta_{goal} = 2°$

Results XI

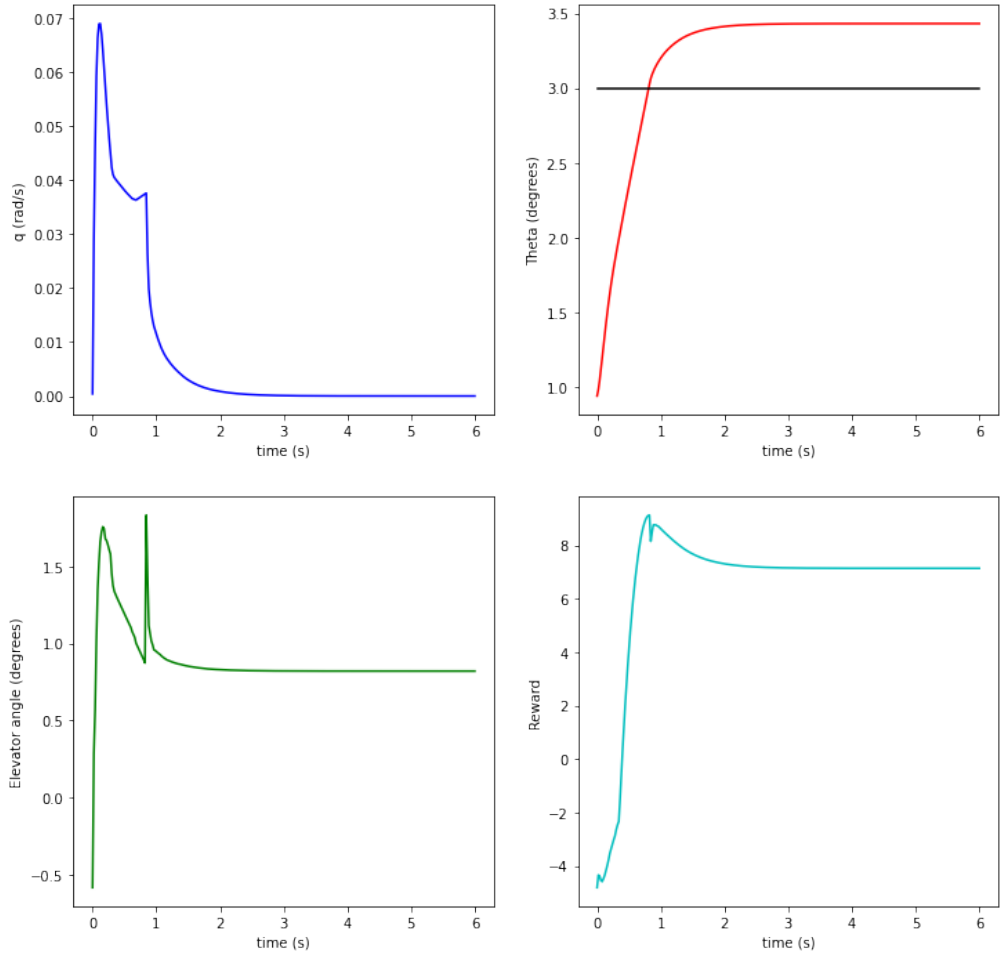


Figure: Results obtained for DDPG agent trained using the state $(q, \Delta\theta, \theta)$ using the second training methodology for a $\theta_{goal} = 3°$

Results XII

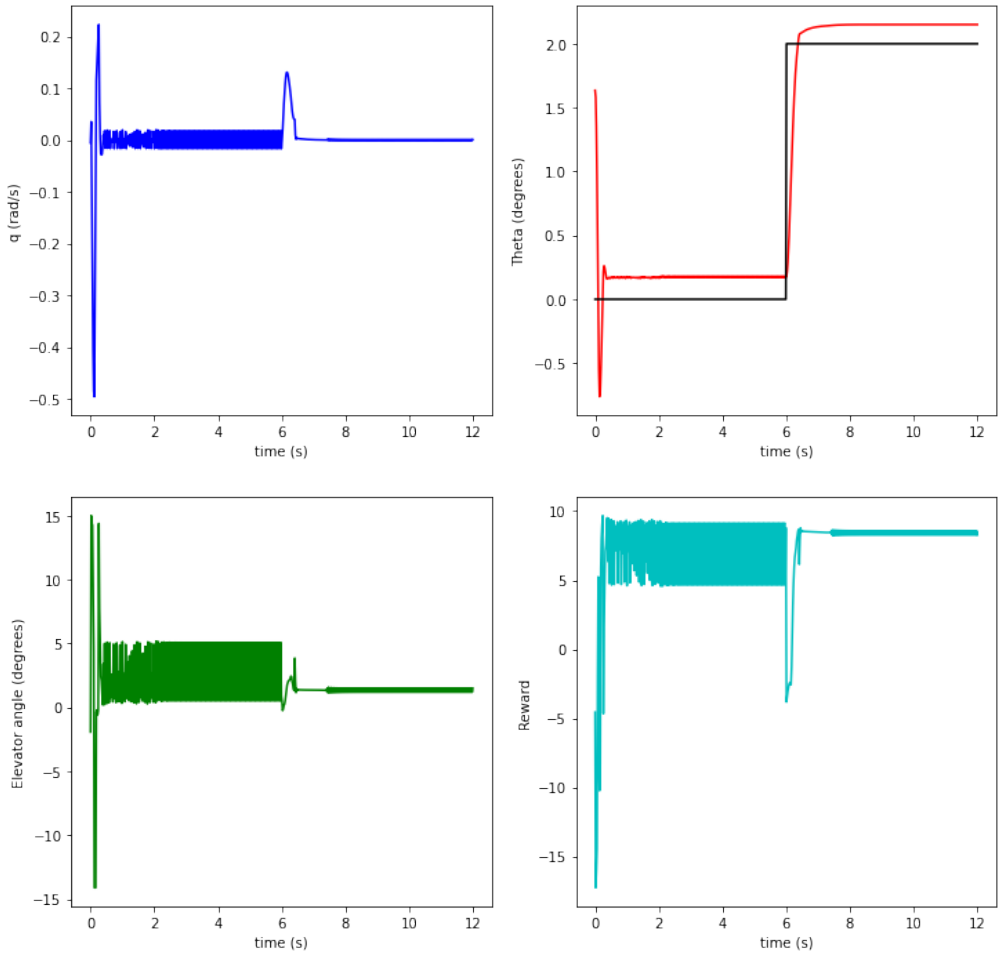


Figure: Results obtained for DDPG agent trained using the state $(q, \Delta\theta, \theta)$ using the second training methodology for a particular profile

Results XIII

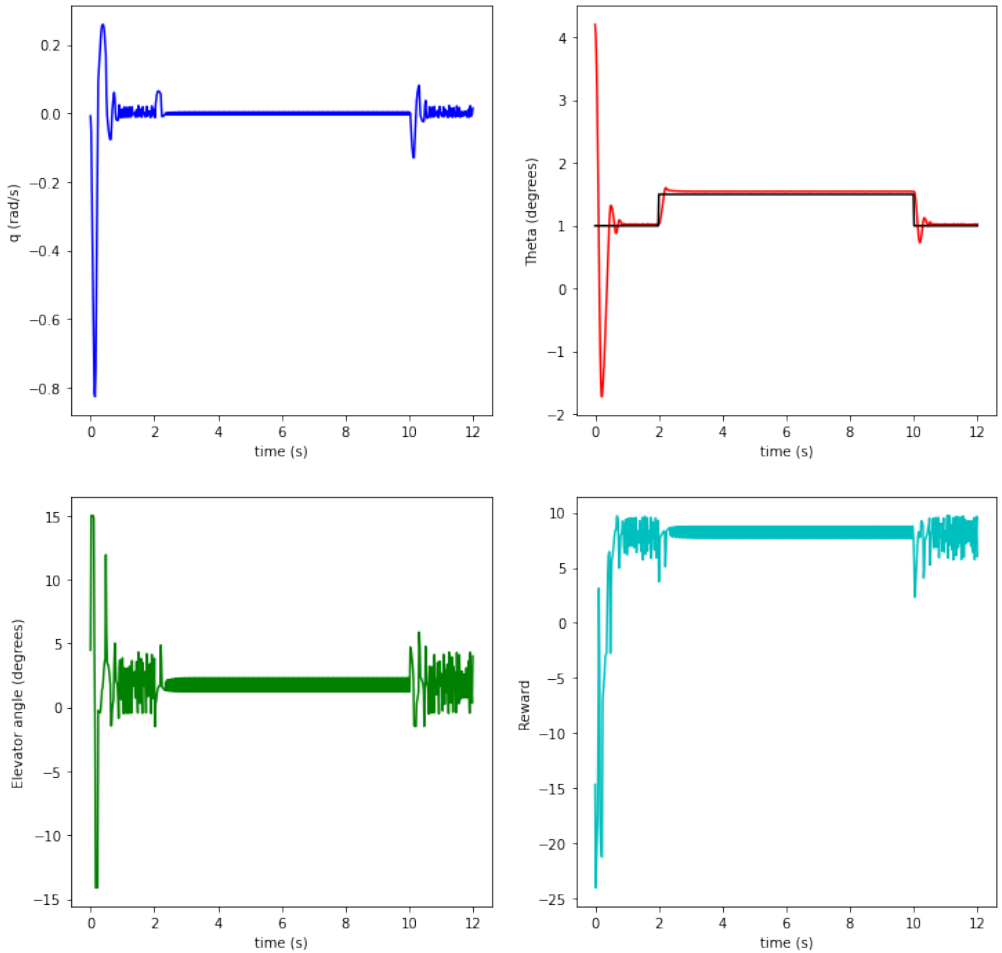


Figure: Results obtained for DDPG agent trained using the state $(q, \Delta\theta, \theta)$ using the second training methodology for a particular profile

Results XIV

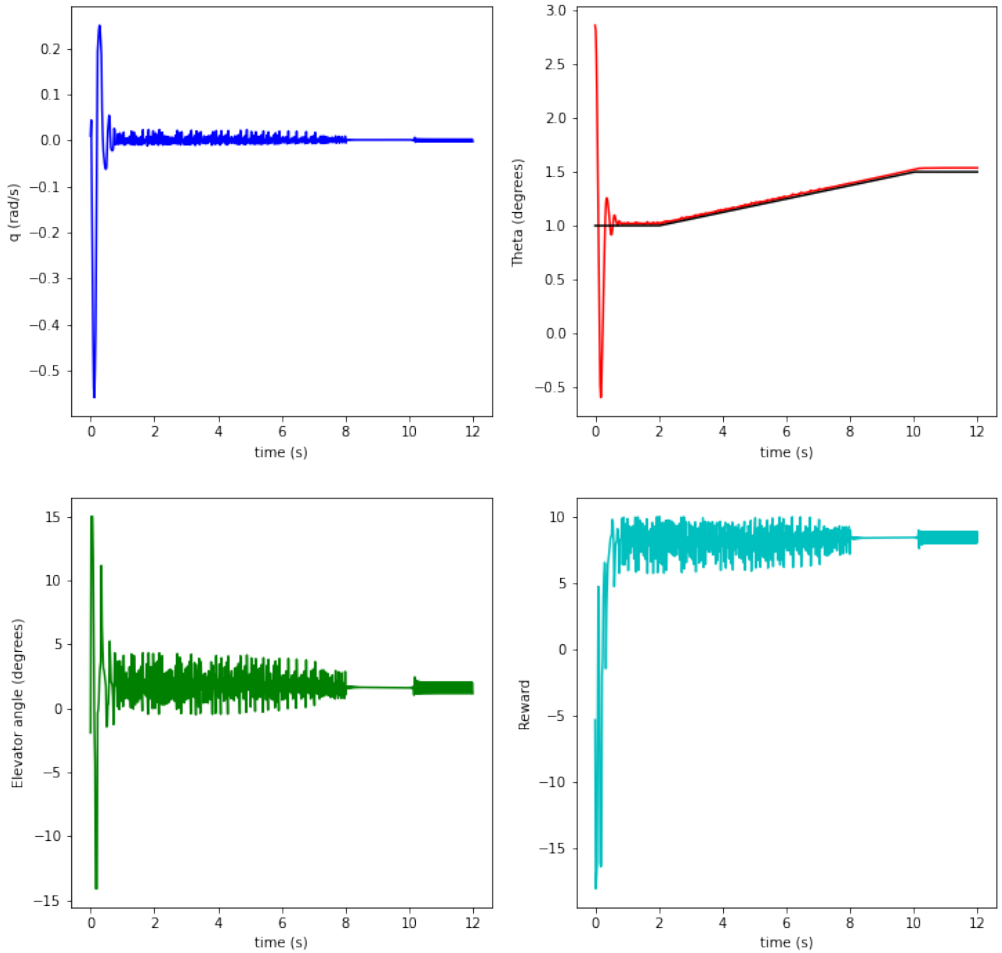


Figure: Results obtained for DDPG agent trained using the state $(q, \Delta\theta, \theta)$ using the second training methodology for a particular profile

Results XV

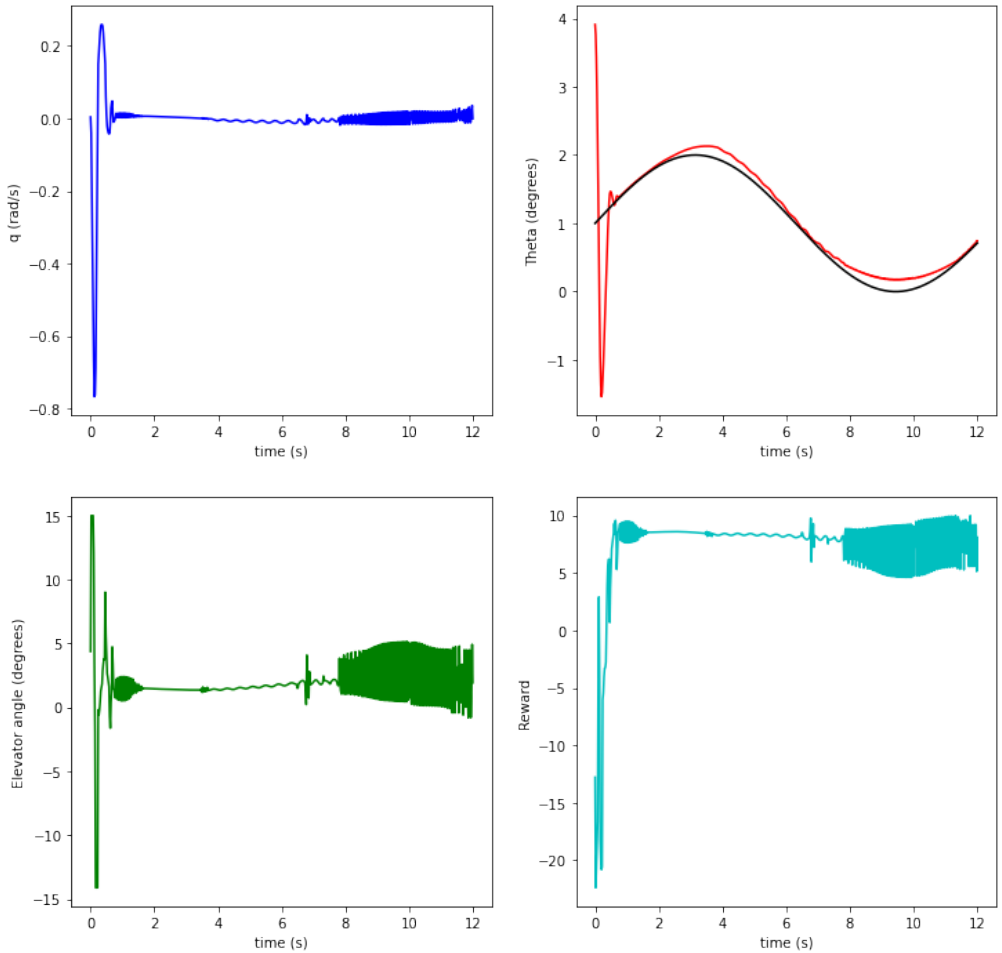


Figure: Results obtained for DDPG agent trained using the state $(q, \Delta\theta, \theta)$ using the second training methodology for a particular profile

Discussions

1. The convergence is smooth in most cases.
2. In between -5 and 5 degrees the steady state error is low with a maximum value of 0.3 degrees
3. The high frequency oscillations are not eliminated but has decreased range as compared to the previous agent.
4. The high frequency oscillations are restricted between 0 and 5 degrees of the elevator.
5. The best possible convergence is for a θ_{goal} of 3 degrees where high frequency oscillations are completely absent from all variables.

Future works

1. Reward function improvement: Optimization approach or other better reward functions.
2. Neural Network shape: Better shape to improve gradient flow.
3. Elevator Modeling: Training using a realistic elevator (saturation and rate saturation) instead of training for a ideal elevator and then bringing the practical limitations. Decreases the action space.
4. Problem representation: δ_e representation in degrees in line with θ representation. Error formulation.
5. Longitudinal Dynamics with 4 states and 2 control inputs.
6. 2 Actor and Critic networks for different ranges of $\Delta\theta$.

THANK YOU

- 2 methodologies for training was tried
  - Method 1 : Fix the θ_{goal} to a constant value and train the reinforcement learning model.
  - Method 2 : Vary the value of θ_{goal} for different episodes.

The training platforms for training of the models.

1. Google Colab GPU/CPU engine
2. Intel Xeon Silver 4114 Processor (10 Core, 2.20 GHz, 13.75 MB L3 Cache), Tesla V100 PCI-E GPU card

The equations hence obtained for longitudinal dynamics are given as Eqns. 1 to 3.

$$m\dot{u} + m(qw - rv) = T_x + X - mgsin\theta \tag{1}$$

$$m\dot{w} + m(pv - qu) = T_z + Z + mgcos\phi cos\theta \tag{2}$$

$$I_{yy}\dot{q} = M + I_{xz}(r^2 - p^2) + (I_{zz} - I_{xx})pr \tag{3}$$

Along with these equations, we also have the attitude equations. 4.

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & tan\theta sin\phi & cos\phi tan\theta \\ 0 & cos\phi & -sin\phi \\ 0 & sec\theta sin\phi & sec\theta cos\phi \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{4}$$

The Aerodynamic model used for the simulation is presented as Eqns. 5 to 10.

$$L = q_\infty S C_L \tag{5}$$

$$D = q_\infty S C_D \tag{6}$$

$$M = q_\infty S \bar{c} C_m \tag{7}$$

$$C_L = C_{L_0} + C_{L_\alpha}\alpha + C_{L_{\delta_e}}\delta_e \tag{8}$$

$$C_D = C_{D_0} + KC_L^2 \tag{9}$$

$$C_m = C_{m_0} + C_{m_\alpha}\alpha + C_{m_{\delta_e}}\delta_e \tag{10}$$

The mass, inertia and geometric properties of the aircraft used for simulations in this project are tabulated in the Tab. 1.

Table: Mass, Inertia and Geometric properties of the aircraft.

Mean Aerodynamic Chord, c	$1.211m$
Wing Span, b	$15.47m$
Aspect Ration, AR	19.9
Wing Area, S	$12.47m^2$
Mass, m	$700kg$
Moment of Inertia, Ixx	$1073kg/m^2$
Moment of Inertia, Iyy	$907kg/m^2$
Moment of Inertia, Izz	$1680kg/m^2$
Moment of Inertia, Ixz	$1144kg/m^2$

Table 2 summarizes the aerodynamic properties of the aircraft.

Table: Aerodynamic properties of the given aircraft.

$C_{D_0} = 0.036$	$C_{L_0} = 0.365$	$C_{m_0} = 0.05$
$C_{D_\alpha} = 0.061$	$C_{L_\alpha} = 5.2$	$C_{m_\alpha} = -0.529$
$e = 0.9$	$C_{L_q} = 17.3$	$C_{m_q} = -5.8$
$C_{D_{\delta_e}} = 0.026$	$C_{L_{\delta_e}} = 0.5$	$C_{m_{\delta_e}} = -1.28$
$C_{Y_0} = 0$	$C_{l_0} = 0$	$C_{n_0} = 0$
$C_{Y_\beta} = -0.531$	$C_{l_\beta} = -0.031$	$C_{n_\beta} = 0.061$
$C_{Y_p} = 0.2$	$C_{l_p} = -0.3251$	$C_{n_p} = -0.015$
$C_{Y_r} = 0.633$	$C_{l_r} = 0.1836$	$C_{n_r} = -0.091$
$C_{Y_{\delta_r}} = 0.15$	$C_{l_{\delta_r}} = 0.005$	$C_{n_{\delta_r}} = -0.049$
$C_{Y_{\delta_a}} = 0$	$C_{l_{\delta_a}} = -0.153$	$C_{n_{\delta_a}} = 0$

Reinforcement learning is a relatively new development in AI and ML. RL agents can now beat humans in games such as chess and Go. As the word suggests it is a way of learning similar to that of

humans who learn by taking actions and analyzing the kind of response he/she gets for the action taken. The major components and their functions in a RL model are:

1. ENVIRONMENT: It is the virtual simulation of any system which acts similar to the real world.
2. STATE (s): Is a representation of the current situation in the environment.
3. AGENT: Agent is the one which observes the state and decides what action to take.
4. ACTION (a): It is decided by the agent and once an action is taken the agent is presented with the new state as a consequence of the action taken according to the rules of the environment and a reward that represents the correctness or some kind of appreciation or criticism about the action taken.
5. REWARD (r): It is rewarded to the agent which then corrects the way in which it calculates the action for a given state based on the reward received.

6. DISCOUNTED SUM OF REWARDS (G) = It can be given by the formula $G_t = \sum_{i=0}^{n} \gamma^i R_{t+i}$. γ is the discount factor which is how much we give importance to the future reward than to the current reward.
7. VALUE FUNCTION: Future rewards from being in a particular state and following a certain policy. It is the maximum of the Expectation of the discounted sum of rewards. It in a way represents the value of a state i.e. moving from a state of lower value to a higher value is beneficial and moving the other way round would not be beneficial to achieve the goal.
8. POLICY: It is the function that the agent uses to map states to actions. Policies can be deterministic or stochastic (probabilistic).

Based on the problem the agent can be modeled in different ways, using ML based feature extractors such as support vector machines belonging to the classical reinforcement learning. The class of reinforcement learning where neural networks are used is known as deep reinforcement learning. CNNs can be used for situations where the state is represented as an image. The problem of flight control agent can be modeled as a neural network itself.
A general flow of the reinforcement learning algorithm is

1. Randomly initialize the policy network.
2. Obtain the state from the environment and take an action. The next action is taken using the policy network and random exploration.
3. Obtain the reward and the next state.
4. Based on the obtained information make relevant changes in the policy network.

The reinforcement learning problems are modelled as a markov decision process, which assumes that the current state is a sufficient statistic of all the history of states and actions. It means that knowing the current state is enough to predict the next state and the information that could be conveyed by the history of all the previous states and actions is conveyed by the current state itself. The value function is given by the bellman equation 11.

$$V(s) = max_a \left(R(s,a) + \gamma \sum s' P(s,a,s') V(s') \right) \qquad (11)$$

Here s' is the state which is reached when the agent takes the action a when at a state s. $P(s,a,s')$ is the probability of moving to the state s' from s by performing the action a. Iteratively substituting $V(s')$ we obtain $V(s)$ as the expected sum of future rewards.
The value defines the maximum of the expected sum of rewards over different actions, thus removing the max part another

function for a state and action pair can be defined known as the quality of the state in Eqn. 12

$$Q(s,a) = R(s,a) + \gamma \sum s' P(s,a,s') V(s')$$

$$Q(s,a) = R(s,a) + \gamma P(s,a,s') max_{a'} Q(s',a') \tag{12}$$

We use this equation to get an estimate of the Q values for the next timestep from the Q values of the previous timestep as in Eqn. **??**

$$Q_t(s,a) = Q_{t-1}(s,a) + \alpha(R(s,a) + \gamma P(s,a,s')$$

$$max_{a'} Q(s',a') - Q_{t-1}(s,a))$$

Maintaining a table for the Q values and updating them after each timestep is easy when the size of the problem is small. For large problems with many possible states, and actions a neural network

is used to map the state to the action Q value pair. An analogy is shown in Fig. 18.

$$Q(s,a) = R(s,a) + \gamma P(s,a,s')max_{a'}Q(s',a') \tag{13}$$

We use this equation to get an estimate of the Q values for the next timestep from the Q values of the previous timestep as in Eqn. **??**

$$Q_t(s,a) = Q_{t-1}(s,a) + \alpha(R(s,a) + \gamma P(s,a,s')$$

$$max_{a'}Q(s',a') - Q_{t-1}(s,a))$$

$$A(s_t,a_t) = Q(s_t,a_t) - V(s_t) \tag{14}$$

$$A(s_t,a_t) = R_{t+1} + \gamma V(s_{t+1}) - V(s_t)$$

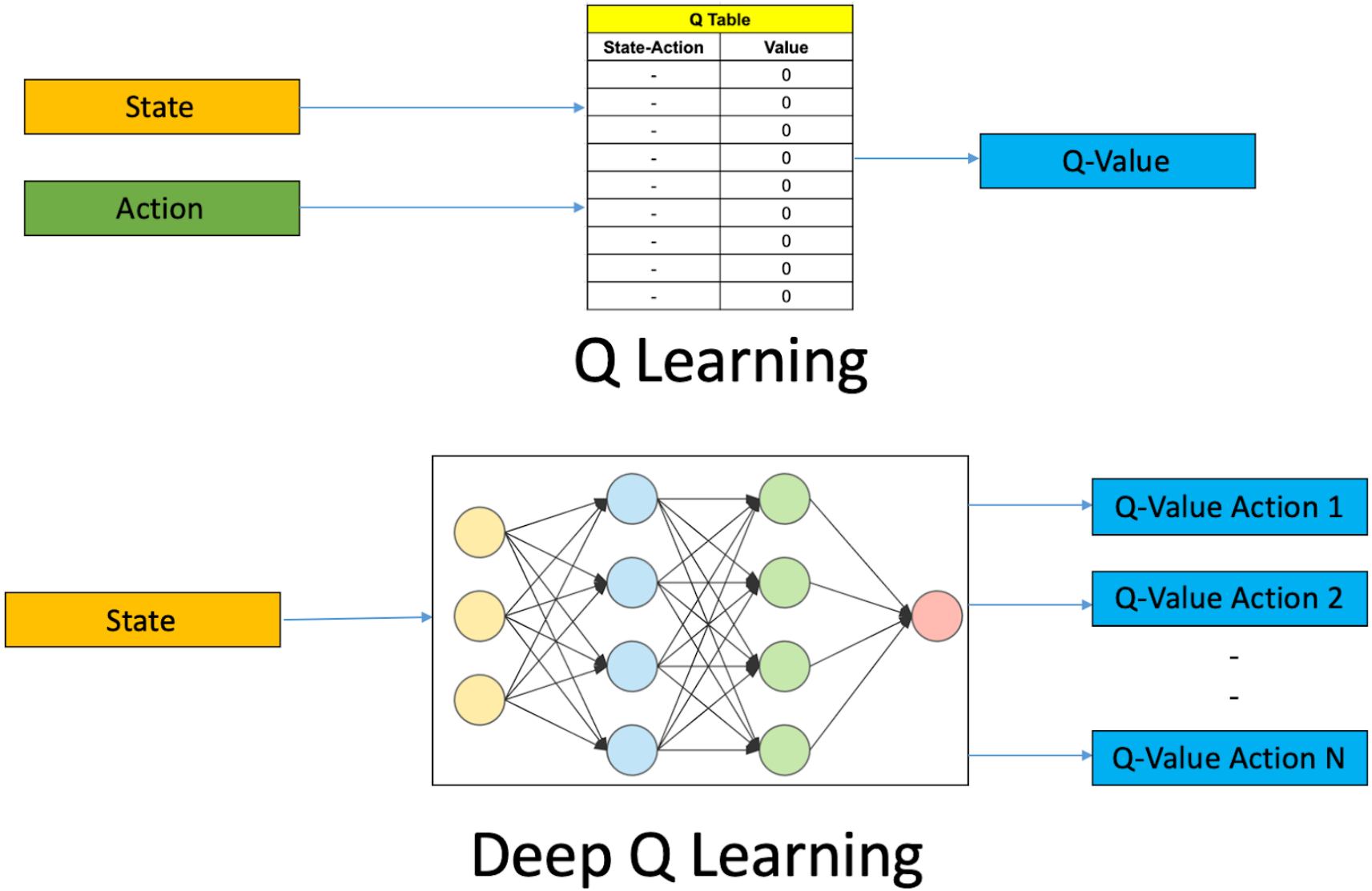
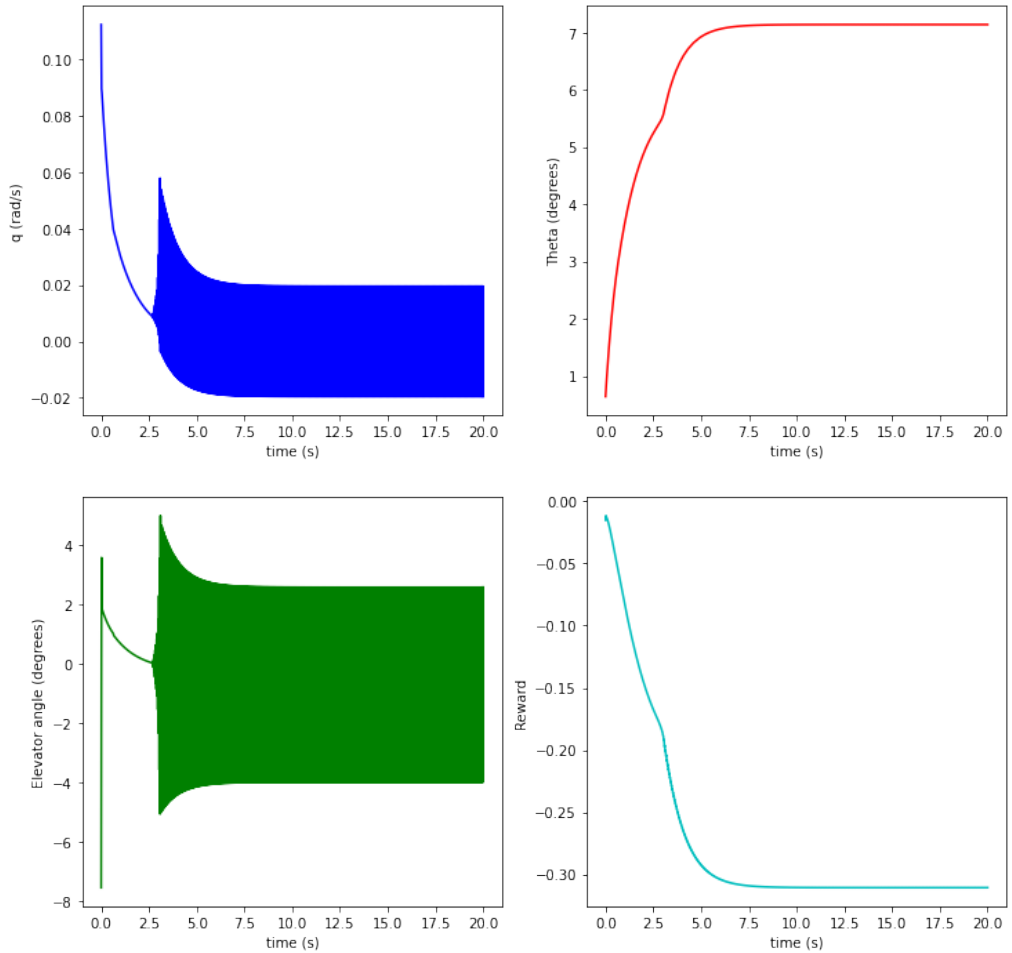


Figure: A representation of DQN

Non Optimal Results I



Non Optimal Results II

Figure: Results obtained for a DDPG agent trained using the state $(q, \Delta\theta, \theta_{goal})$ using the second training methodology for a θ_{goal} of 4°

Non Optimal Results III

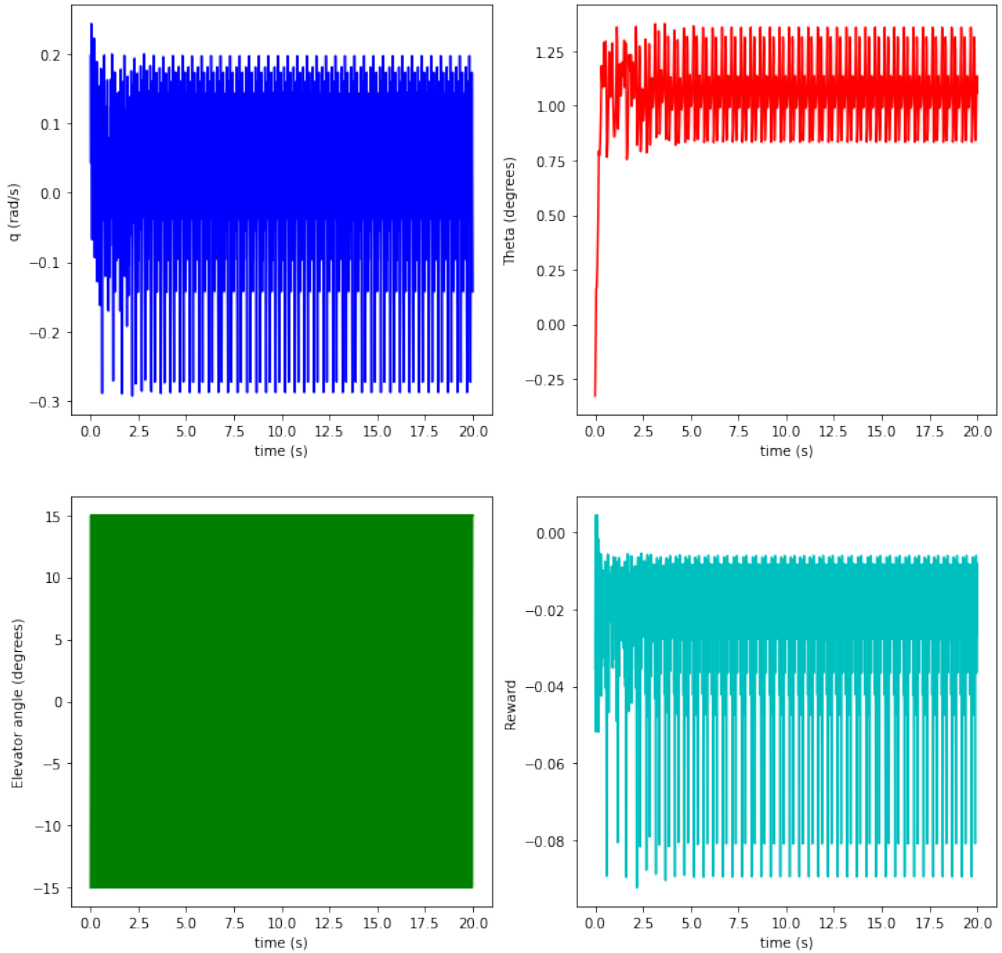


Figure: Results obtained for a DDPG agent trained using the state $(q, \Delta\theta, \theta_{goal})$ using the second training methodology for a θ_{goal} of 0°

Non Optimal Results IV

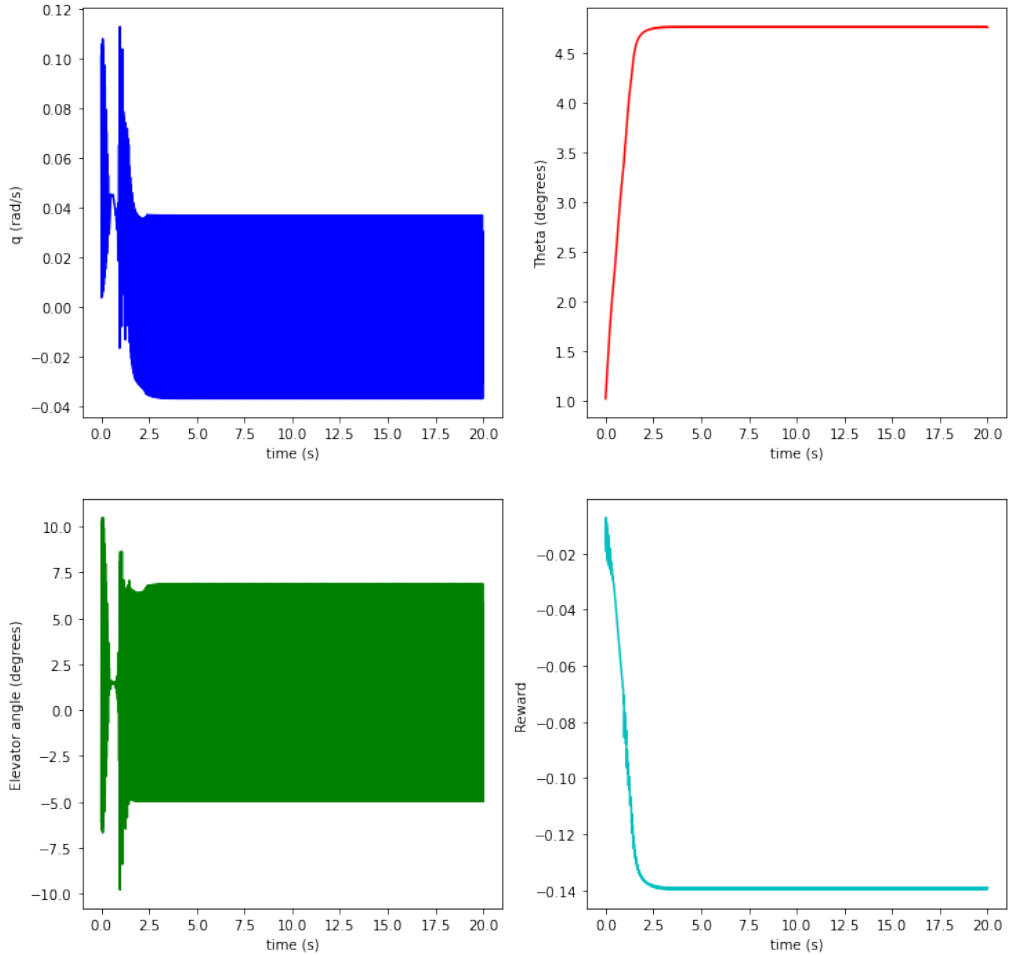


Figure: Results obtained for a DDPG agent trained using the state $(q, \Delta\theta, \theta_{goal})$ using the second training methodology for a θ_{goal} of 3°

Non Optimal Results V

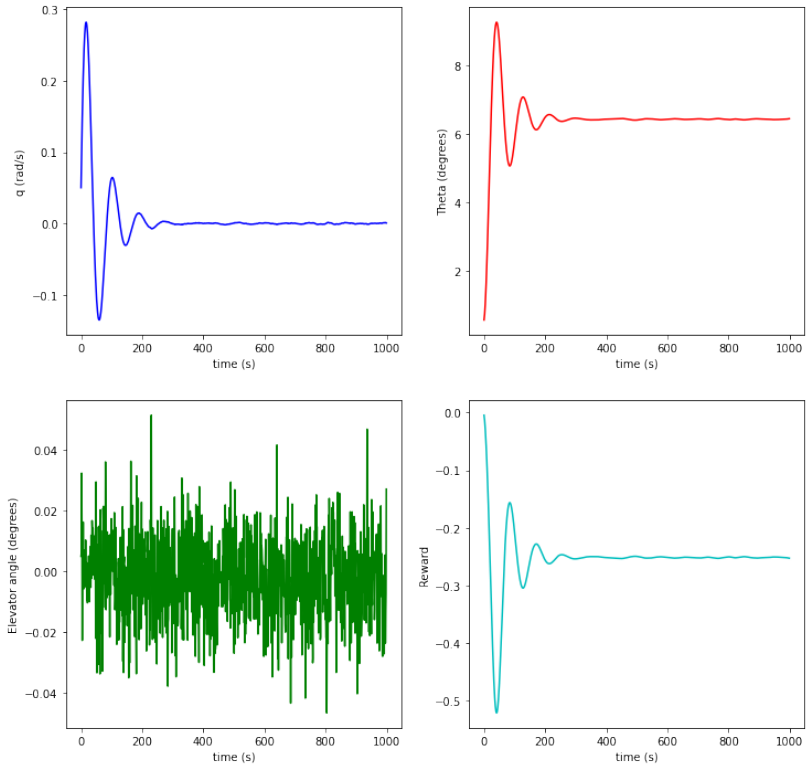


Figure: Results obtained for A3C agent trained using the state $(q, \Delta\theta, \theta_{goal})$ using the second training methodology with the elevator angle shown in radians for a θ_{goal} of 4°

Non Optimal Results VI

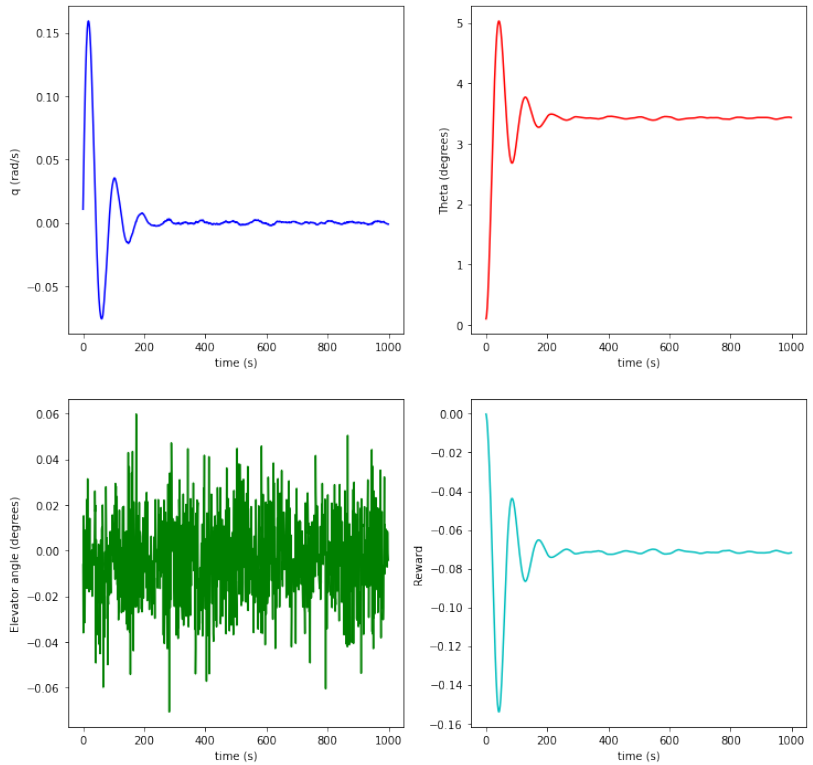


Figure: Results obtained for A3C agent trained using the state $(q, \Delta\theta, \theta_{goal})$ using the second training methodology with the elevator angle shown in radians for a θ_{goal} of 2°

Non Optimal Results VII

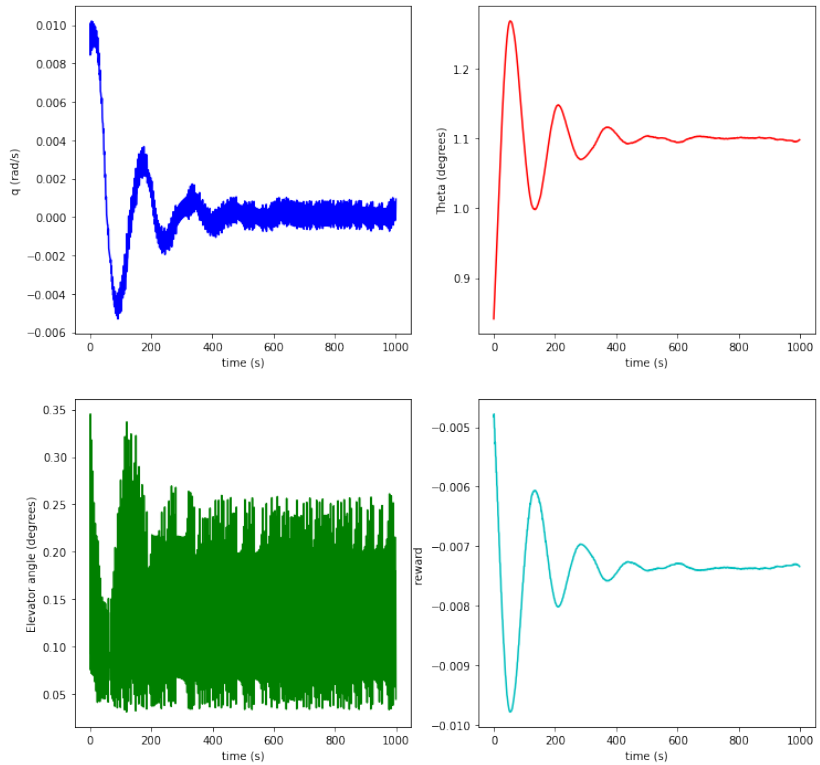


Figure: Results obtained for SAC agent trained using the state $(q, \Delta\theta, \theta_{goal})$ using the second training methodology with the elevator angle shown in radians for a θ_{goal} of 1°

Non Optimal Results VIII

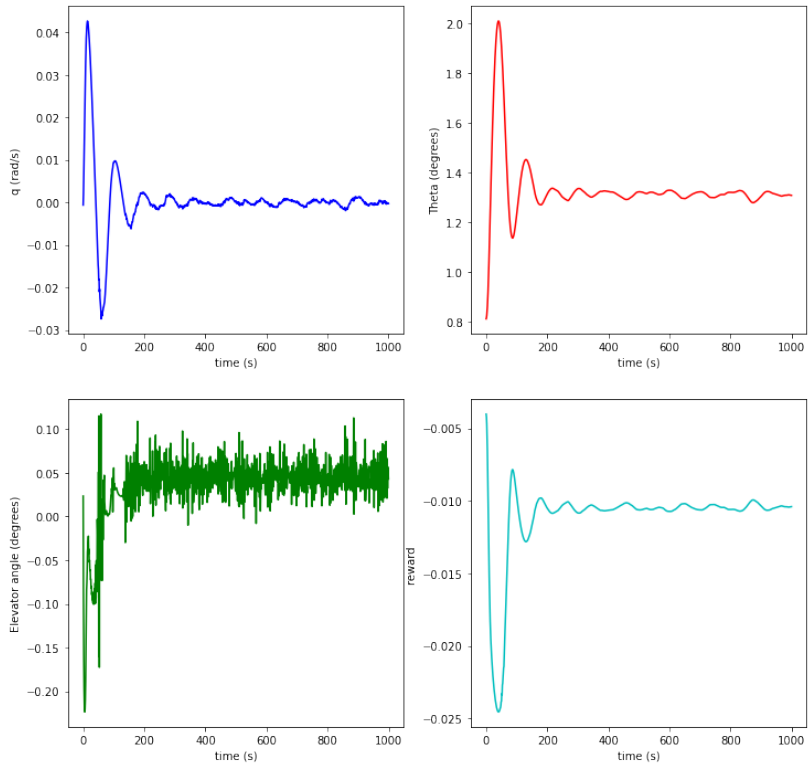


Figure: Results obtained for SAC agent trained using the state $(q, \Delta\theta, \theta_{goal})$ using the second training methodology with the elevator angle shown in radians for a θ_{goal} of 1°

Non Optimal Results IX

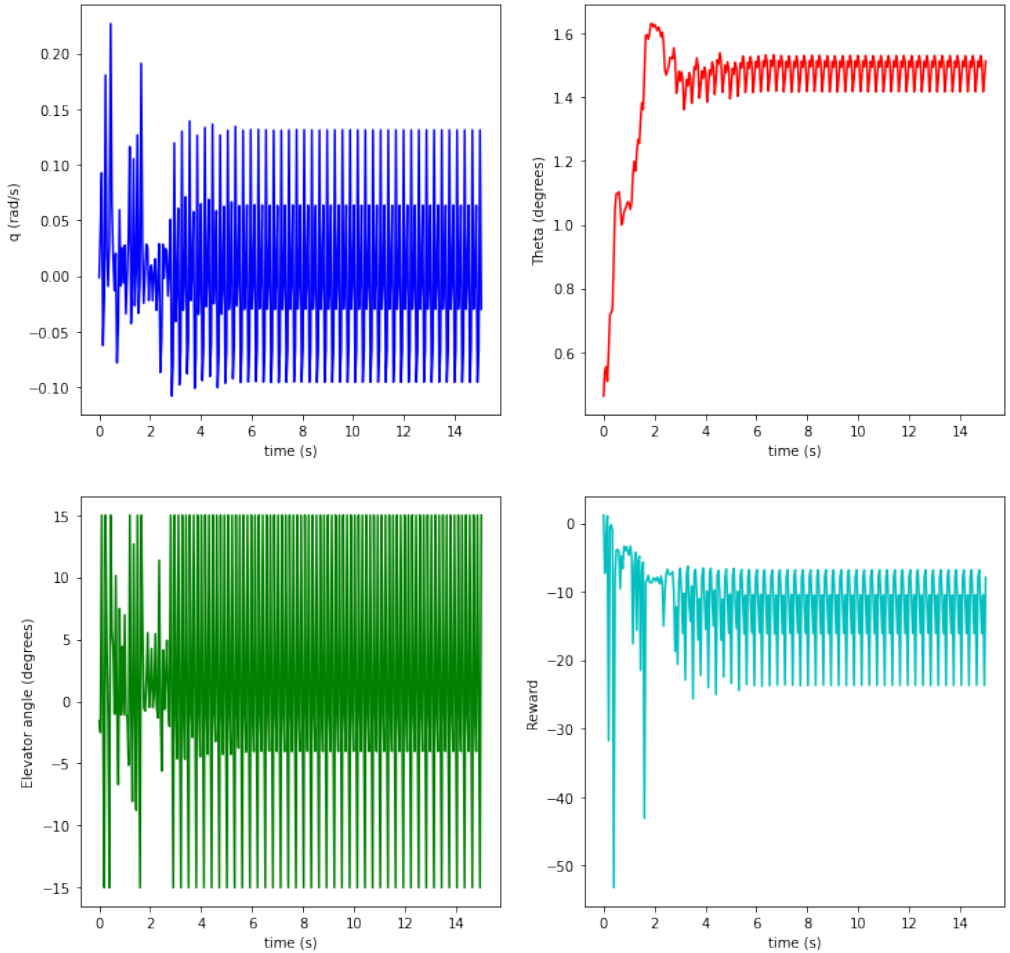


Figure: Results obtained for SAC agent trained using the state $(q, \Delta\theta, \dot{q})$ using the second training methodology for a $\theta_{goal} = 0°$

Non Optimal Results X

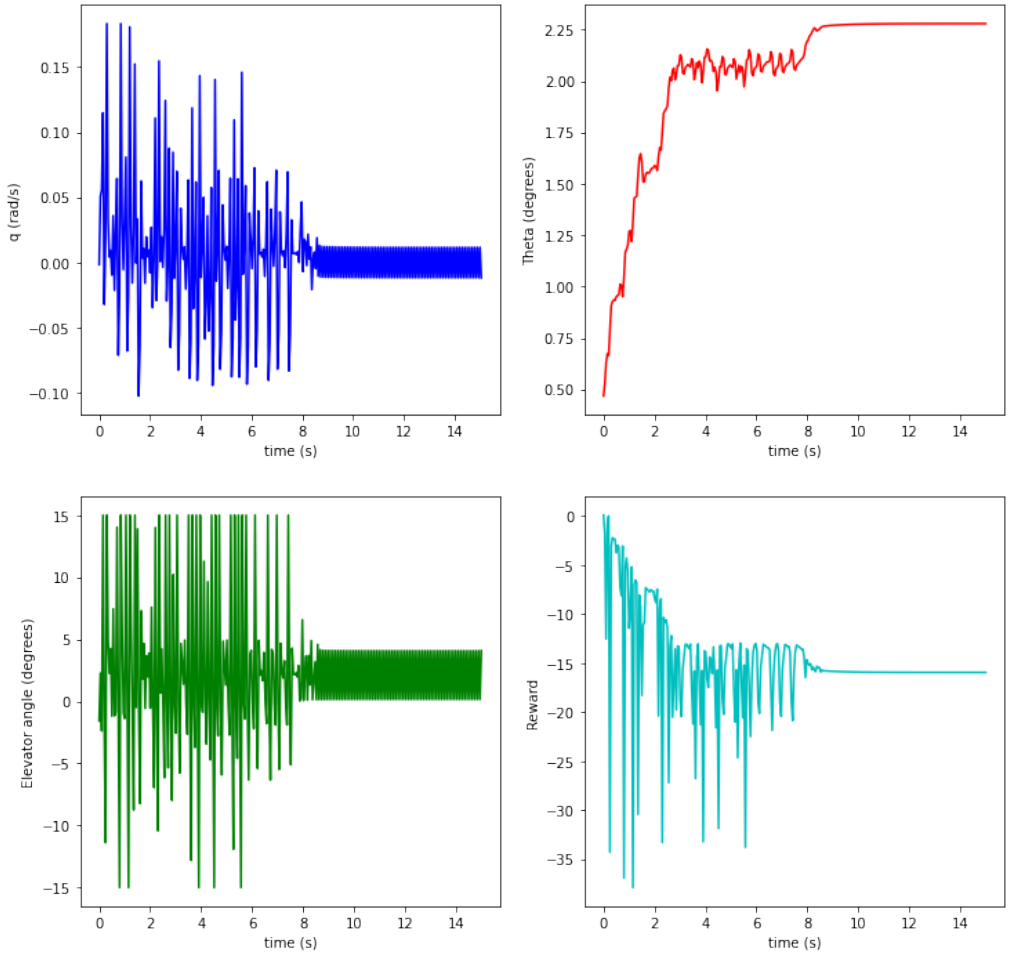


Figure: Results obtained for SAC agent trained using the state $(q, \Delta\theta, \dot{q})$ using the second training methodology for a $\theta_{goal} = 2°$

Non Optimal Results XI

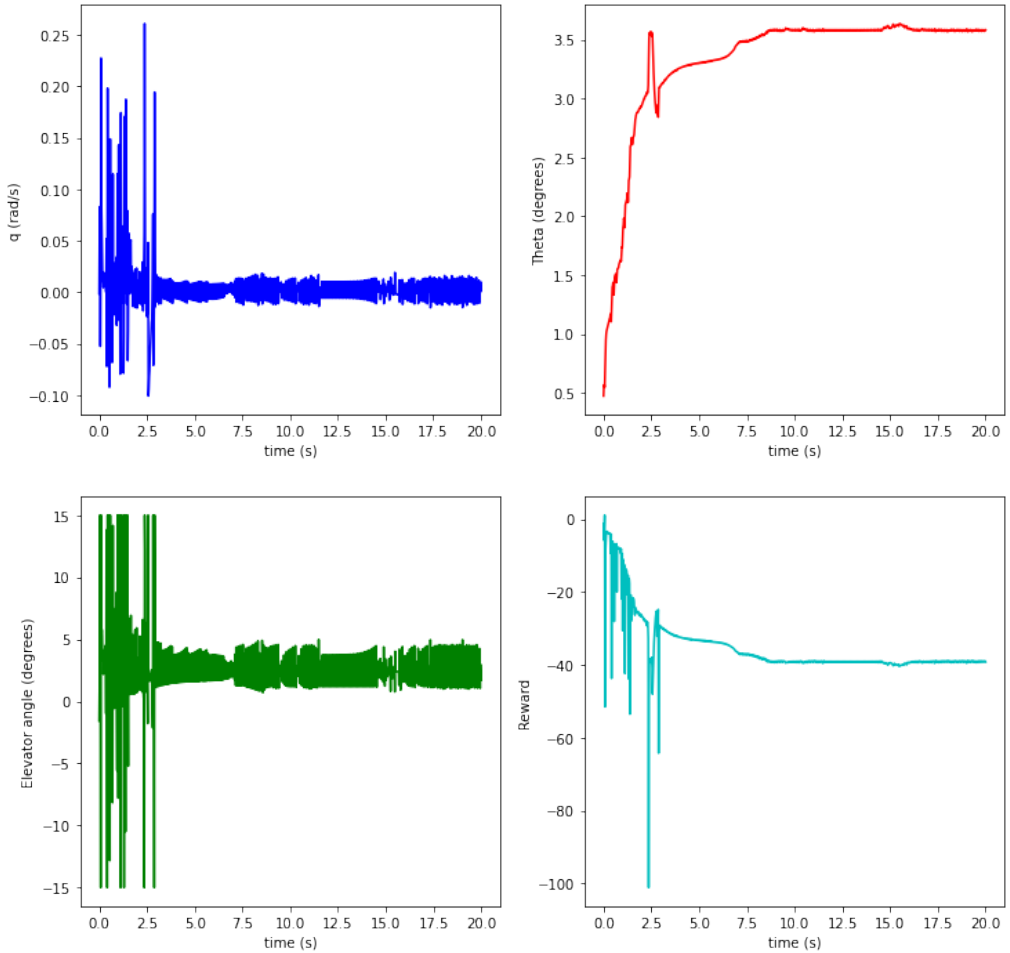


Figure: Results obtained for SAC agent trained using the state $(q, \Delta\theta, \dot{q})$ using the second training methodology for a $\theta_{goal} = 4°$

Non Optimal Results XII

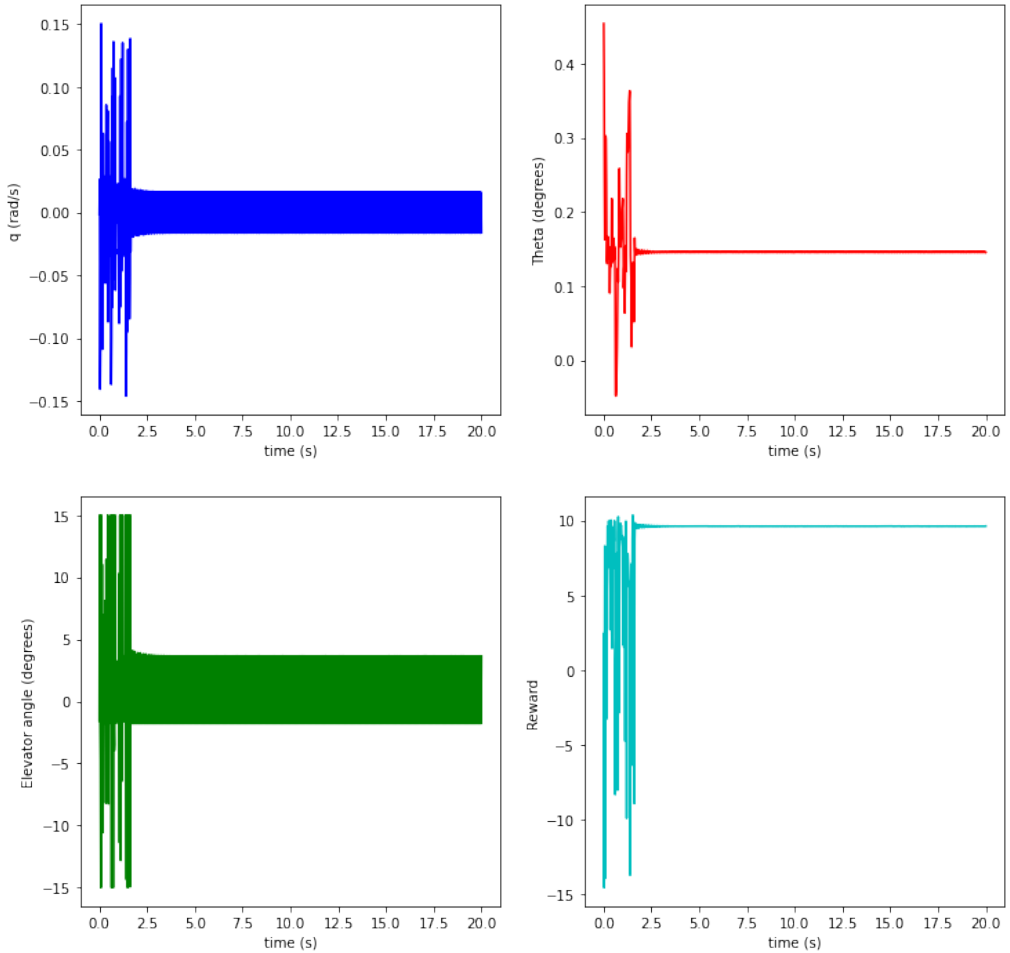


Figure: Results obtained for SAC agent trained using the state $(q, \Delta\theta, \dot{q})$ using the second training methodology for a $\theta_{goal} = -3°$

Non Optimal Results XIII

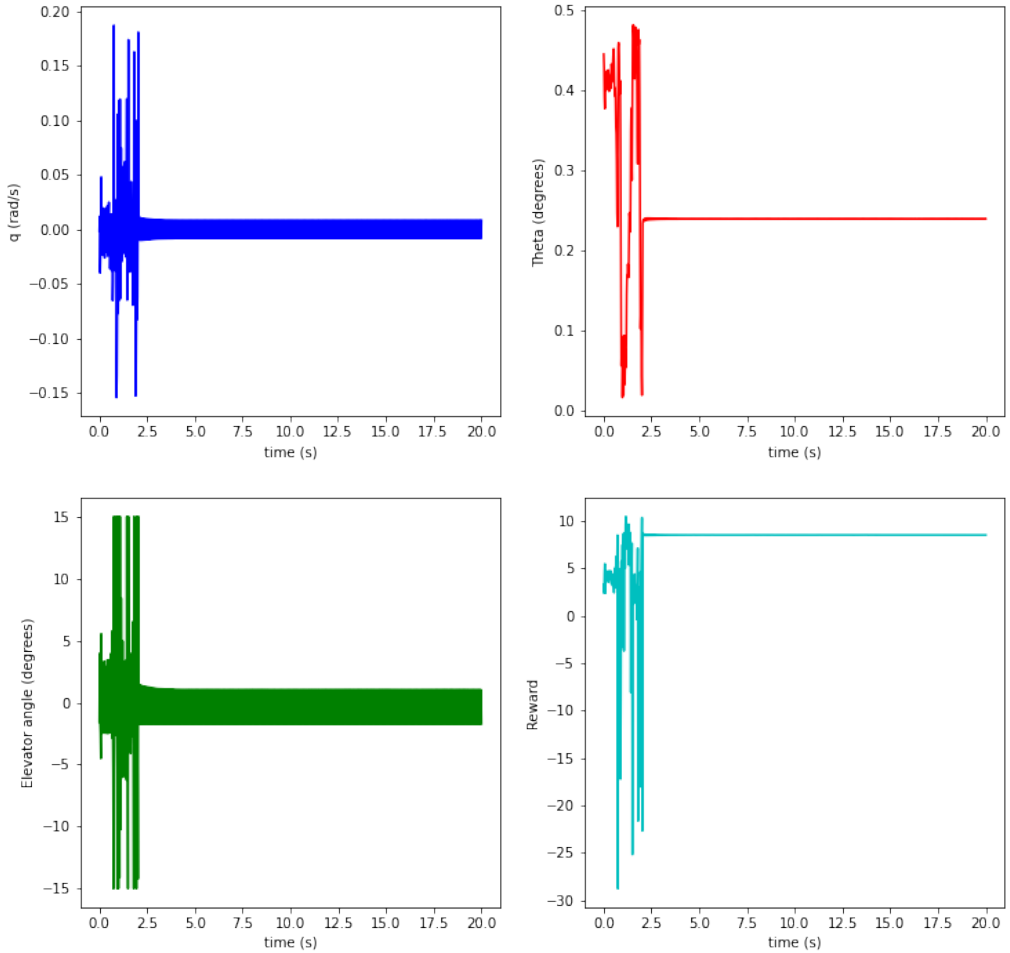


Figure: Results obtained for SAC agent trained using the state $(q, \Delta\theta, \dot{q})$ using the second training methodology for a $\theta_{goal} = -6°$

Non Optimal Results XIV

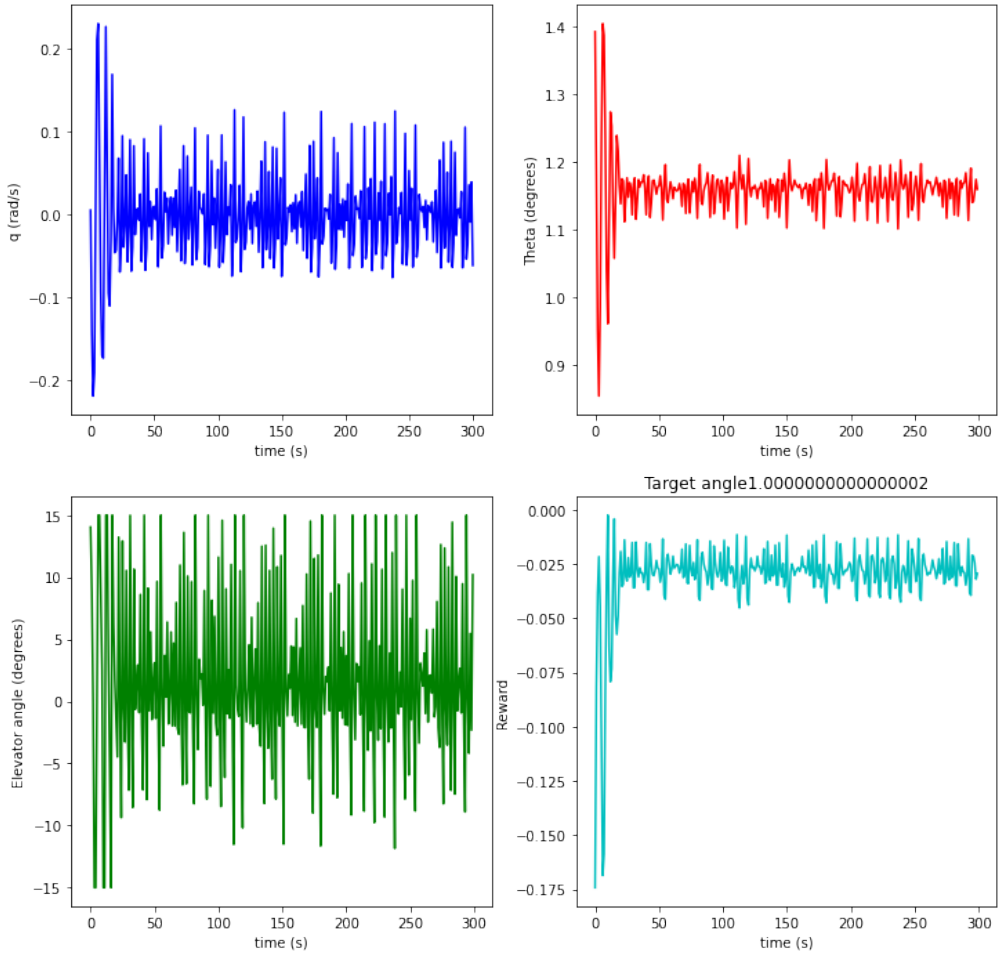


Figure: Results obtained for DDPG agent trained using the state $(q, \Delta\theta, \theta)$ using the second training methodology for a $\theta_{goal} = 1°$

Non Optimal Results XV

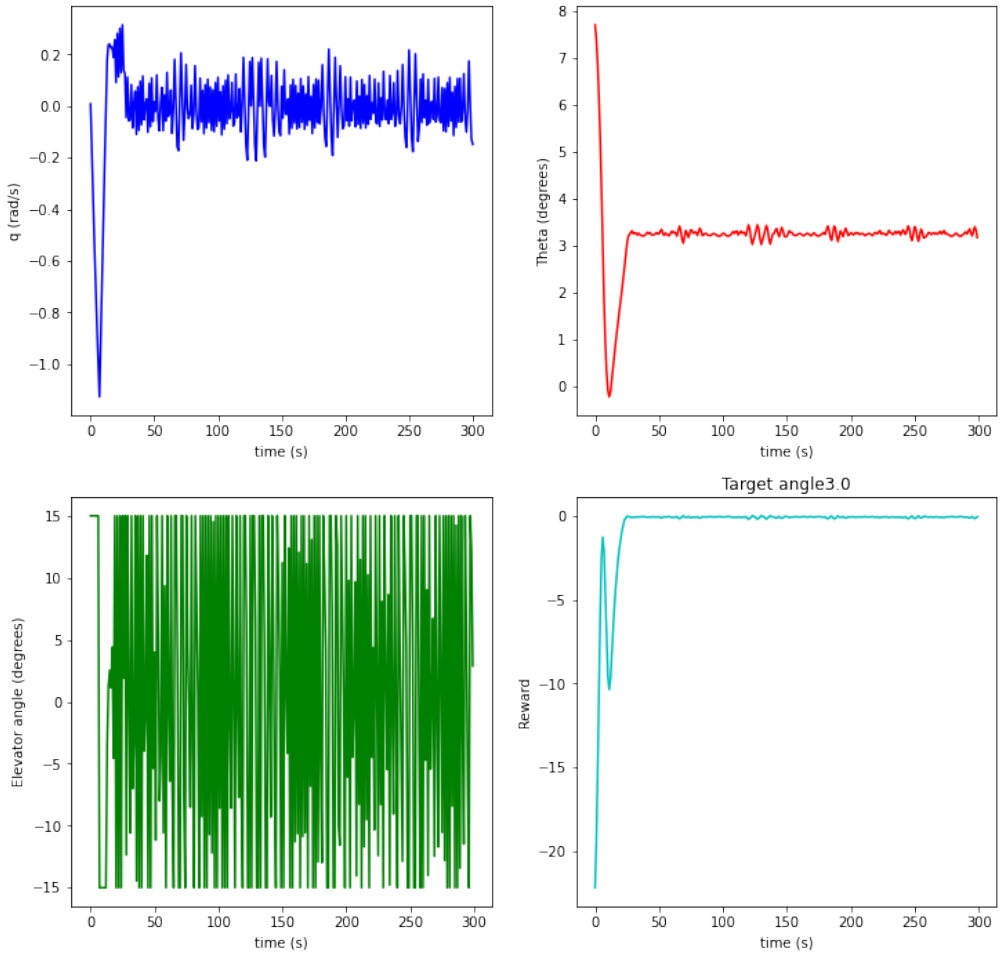


Figure: Results obtained for DDPG agent trained using the state $(q, \Delta\theta, \theta)$ using the second training methodology for a $\theta_{goal} = 3°$

Non Optimal Results XVI

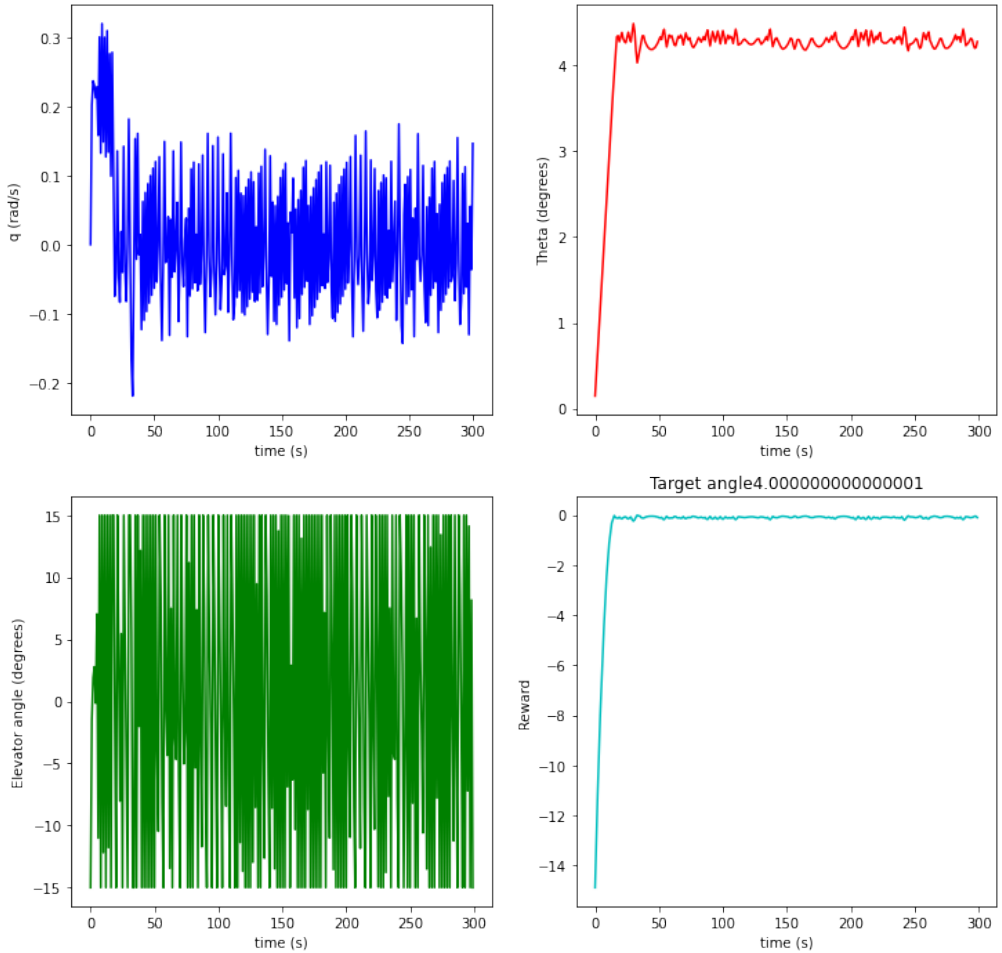


Figure: Results obtained for DDPG agent trained using the state $(q, \Delta\theta, \theta)$ using the second training methodology for a $\theta_{goal} = 4°$

Non Optimal Results XVII

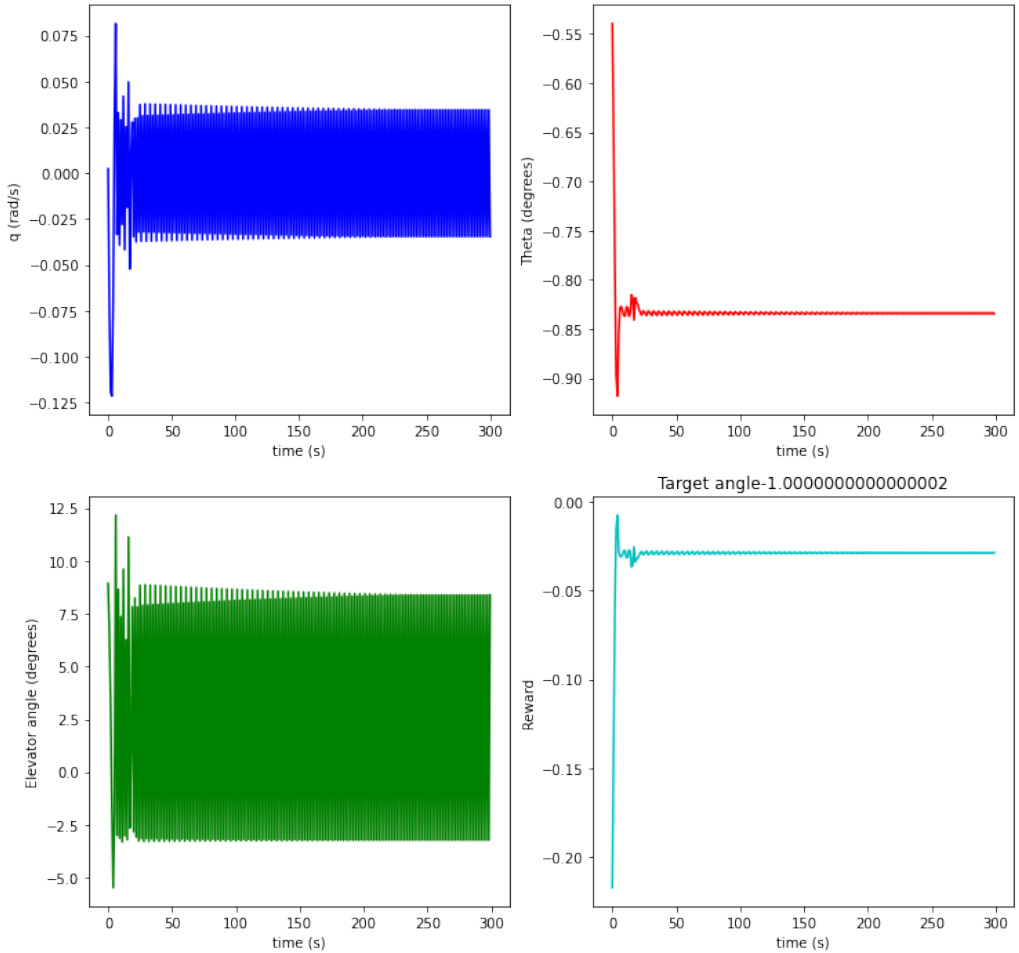


Figure: Results obtained for DDPG agent trained using the state $(q, \Delta\theta, \theta)$ using the second training methodology for a $\theta_{goal} = -1°$

Non Optimal Results XVIII

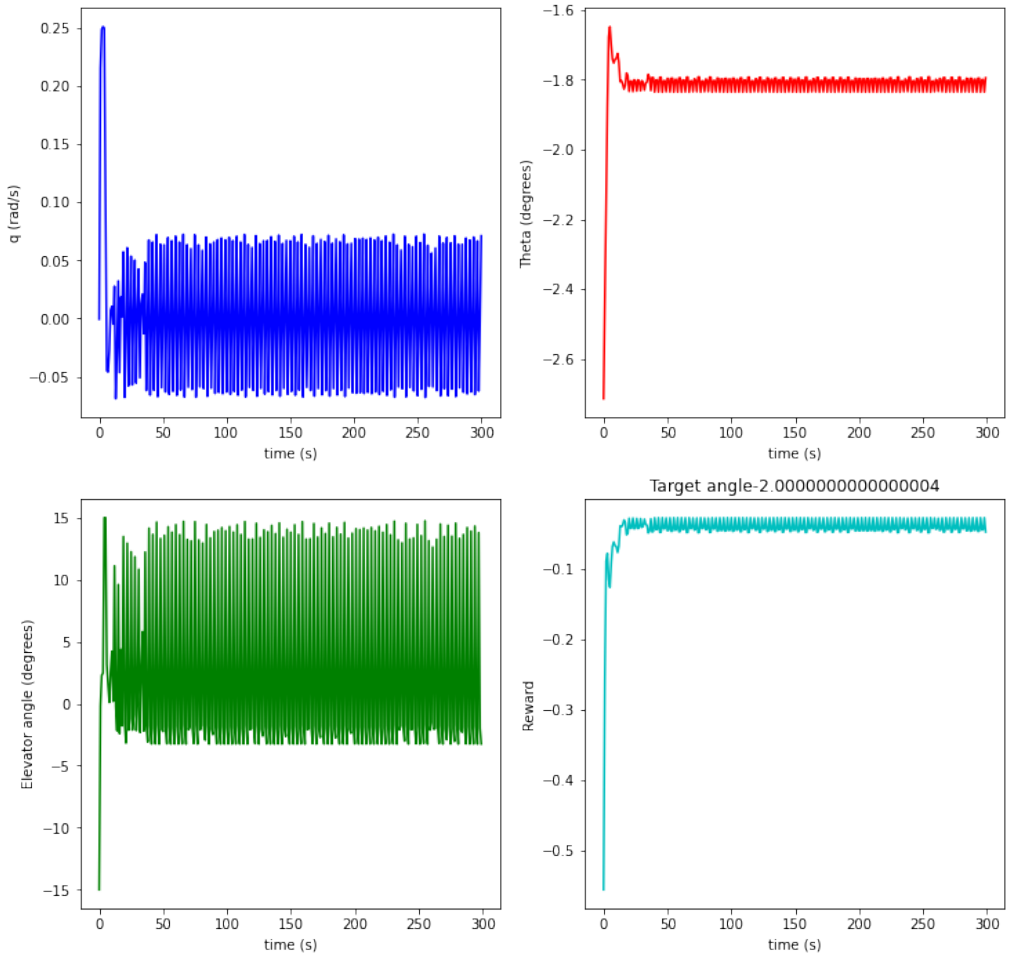


Figure: Results obtained for DDPG agent trained using the state $(q, \Delta\theta, \theta)$ using the second training methodology for a $\theta_{goal} = -2°$

Non Optimal Results XIX

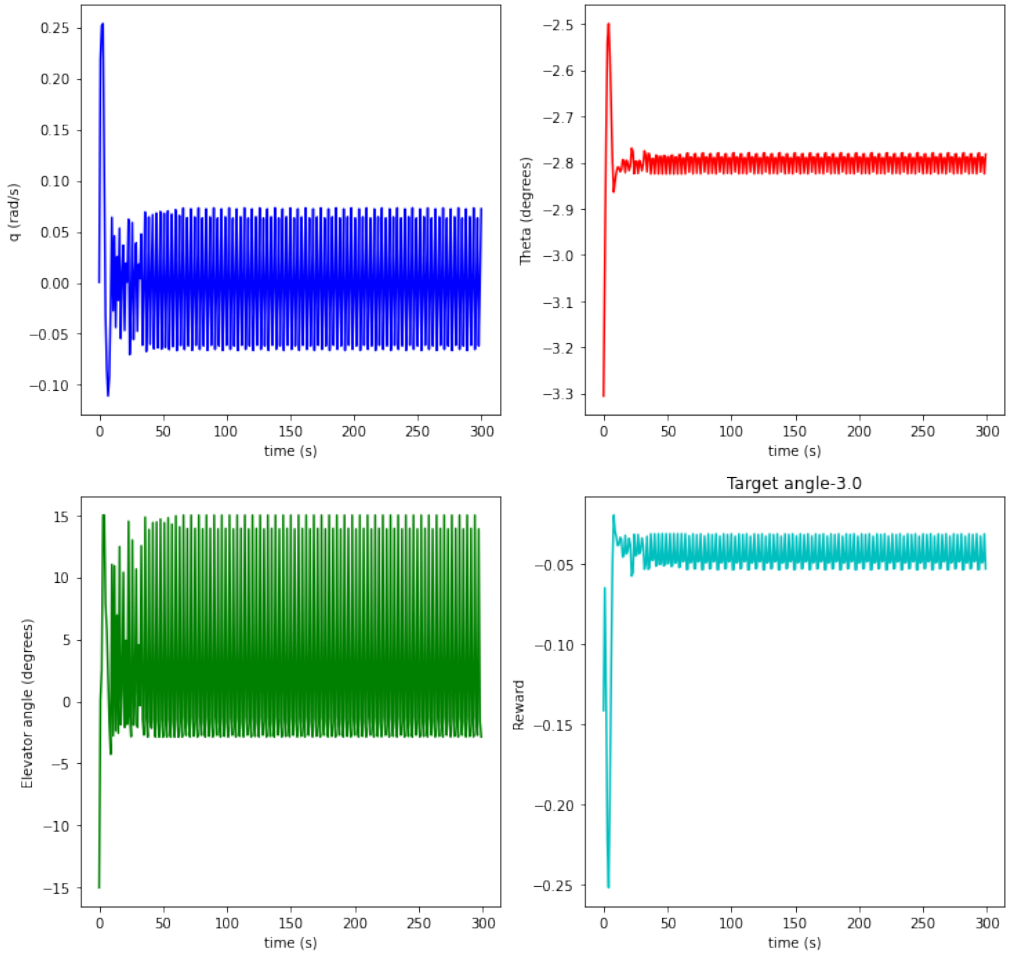


Figure: Results obtained for DDPG agent trained using the state $(q, \Delta\theta, \theta)$ using the second training methodology for a $\theta_{goal} = -3°$

Non Optimal Results XX

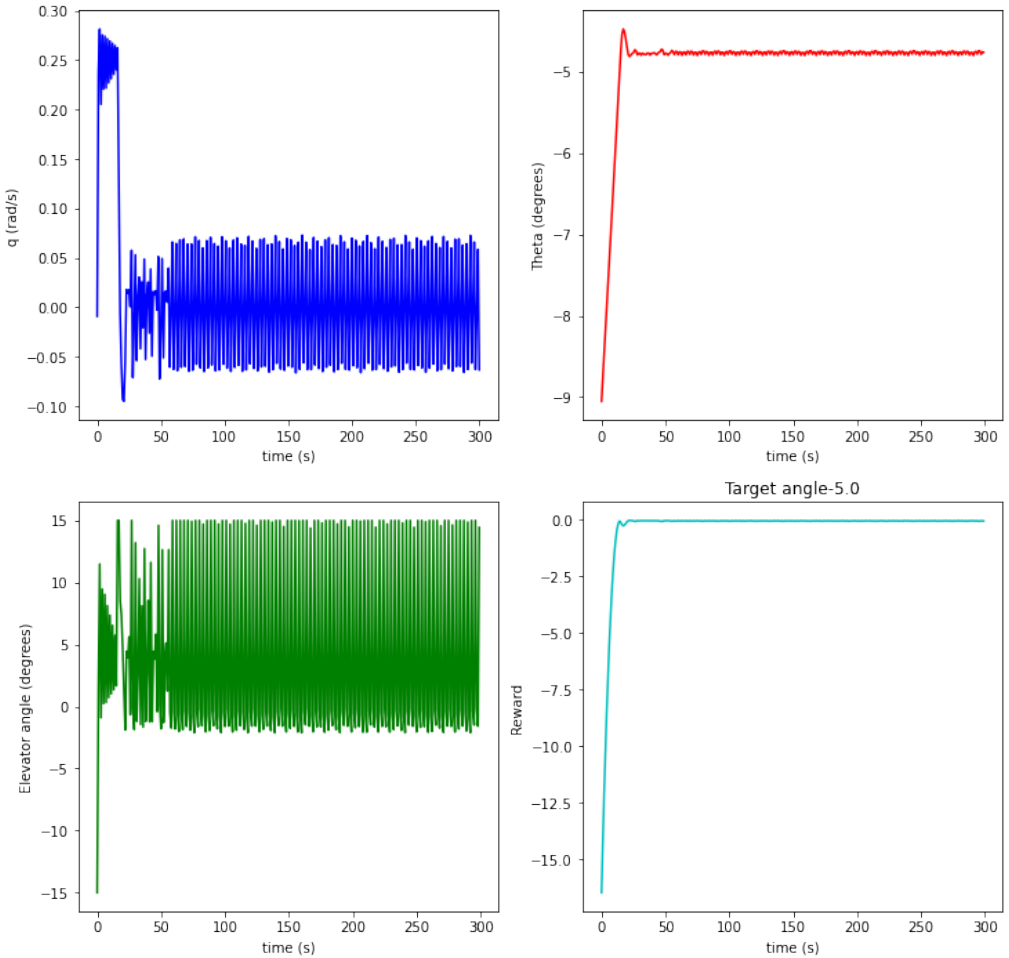


Figure: Results obtained for DDPG agent trained using the state $(q, \Delta\theta, \theta)$ using the second training methodology for a $\theta_{goal} = -5°$