**Title: Application of Reinforcement learning to select design matrices in LQR for Drone control**

**Contents**

Title Page (see above)

Declaration (see above)

Abstract

Acknowledgements

List of Contents (sections and sub-section with page numbers)

List of Symbols

Note a list of Figures and Tables is NOT required. Neither is it necessary to include units in your nomenclature.

Main Body of work

This will be split into any number of sections, with appropriate titles and numbered 2, 3 etc.

1. **Introduction**
   1. Motivation: Linear Quadratic regulator(LQR) and difficulty is choosing Q and R matrices (design matrices).
   2. Application: Drones! Benefits of faster response.
2. **Literature survey**: *Analyse advantages and disadvantages of literature in each section below.*
   1. LQR: its application to various systems (including drones).
      1. Survey of methods for choosing design matrices in LQR.
   * Example: Bryson’s rule, reinforcement learning (RL).
   * Note: Mention that using LQR with RL still guarantees optimality whereas using standalone RL agent to produce control inputs provides no such guarantee
     1. Methods for combining LQR with Proportional Integral Derivative (PID) control.
   * This should include Integral Square Error (ISE) optimization
   1. Reinforcement Learning for continuous action space
   * Include RL algorithm used in code: <https://arxiv.org/abs/1509.02971>
3. **Theory**
   1. Quadrotor dynamics, state space equation
   2. LQR
   3. PID control
      1. Deriving PID gains using ISE optimization
   4. Reinforcement Learning background
4. **Methodology**
   1. Control method for drone: LQR/ LQR+ PD control
   2. Bryson’s rule for selecting design matrices
      1. Changing all diagonal elements
      2. Changing only 4 diagonal elements.
   3. Reinforcement learning for selecting design matrices
      1. Task formulation: state, transition, action and reward in our context. (mention we are tuning only 4 diagonal elements)
      2. Actor, Critic neural network architecture. More details about the implementation like batch normalization.
5. **Numerical Experiments**

Few words about Numerical Experiments setup: Training and test conditions and how set point or trajectory is generated.

* 1. Test 1: Reaching a set point.

Results should include:

1. Training curve for RL agent: mean reward, reward per episode, number of failures, average length of episode.
2. Number of failures while training with RL agent. (the drone should be able to reach the set point more consistently as the training episodes increase)
3. ISE of all 4 methods over 100 random tests. Report number of failures when testing with RL agent.
4. Sample plot with all 4 methods.
   1. Test 2: Following a trajectory
5. Training curve for RL agent.
6. Number of failures while training with RL agent. (the drone should be able to reach the desired goal more consistently as the training episodes increase)
7. ISE of all 4 methods over 100 random tests. Report number of failures when testing with RL agent.
8. Sample plot with all 4 methods.
   1. Test 3: TODO: any additional tests.

This could include:

1. giving extra features like next two waypoints to the RL agent.
2. Different reward functions:
   1. Penalty due to limits on maximum input values (due to limit on maximum torque generated by motors).
   2. Penalty due to singularity of Q matrix: The Q matrix in LQR should well behaved.
3. Robustness of RL agent to noise in sensor measurements or model.
4. Robustness and response of RL agent to sudden change in trajectory due to obstacle.
5. **Conclusion**

**References** Not a numbered section

**Appendices** (designated Appendix A, B etc. with subsections A1, A2, B1 if appropriate)

**5. Numerical Experiments**

The learning agent’s ability to reach the desired set point and follow a trajectory was tested on MATLAB. This section presents a discussion on the results obtained for both the above cases. We begin by describing the notation and quantitative metrics used to compare different controllers.

**Notation:** The state of the drone will be represented using a 12-dimensional vector, . The state vector will have the following structure,

A single element from the state vector will be indexed as . For example, the velocity in the direction can be notated as . A waypoint is an intermediate state configuration the drone should reach while following a trajectory. The waypoint on a trajectory will be notated using . A notation similar to the state vector is followed to notate individual elements in the waypoint. For example, the position of a waypoint will be shown as .

Four quantitative metrics will be used to compare the performance of different controllers, rise time, settling time, integral square error (ISE) and time of flight. Rise time is the time the quadrotor takes to go from 10% to 90% of its goal position. Settling time is the time the quadrotor takes to reach and stay within 2% of its steady state position. The time of flight is the total time the quadrotor takes to reach its goal position from the initial state. ISE is formed by integrating the square of the error (difference between desired and current state of the drone) over its total time of flight.

* 1. **Test 1: Reaching a set point.**

Experiments in this section were designed to answer the following questions:

1. Is the learning-based approach able to reach a random desired set point?
2. How does the learning-based approach compare to the following three benchmarks:
   1. LQR with identity Q and R matrices.
   2. LQR with Bryson’s rule to tune Q and R matrices.
   3. LQR and PID control with Bryson’s rule to tune Q and R matrices in the LQR.

**Overview of training process.**

In each training episode, the MATLAB RL environment described in section 4.3 generates a random set point for the quadrotor to reach. This new desired set point has a random value for x, y and z coordinates between the closed interval [-1m, 1m]. The desired angular displacements and velocities are set to zero. Overall, the desired state vector has the following structure:

The quadrotor starts from an initial state of zero position and velocity. In each iteration of the episode, the RL agent receives the current state of the quadrotor and outputs the four diagonal elements of the Q matrix as described in section 4.3.1. With the updated Q matrix, the controller outputs the thrust for each rotor which drives the quadrotor’s state towards the desired state.

**Results**

*Training*

The RL agent was trained for 100 episodes, and the reward per episode was recorded. It is worth noting that a lower reward does not necessarily represent a better trajectory as each episode has a different desired state. The average reward till the current episode is a more robust measure to monitor the agent's learning. Figure 1 shows these two quantities after each episode.



Figure 1. Training progress of the RL agent to reach a desired set point

The RL agent collects better rewards as the training progresses (Figure 1). As the agent learns a meaningful Q function and policy, it is expected that the quadrotor should be able to reach the desired state more often and in less number of iterations. Figure 2 shows the number of iterations taken by the drone to reach the desired set point in a given episode. The number of iterations required reduces to lower values as the training progresses.



Figure 2. Number of iterations taken by RL agent to reach the desired state per episode

*Testing*

After training the agent for 100 episodes, it was tested on 50 new random set points. The agent successfully reached all the test set points within the allocated time of 10 seconds. Figure 3 shows the trajectory taken by different controllers for one of these test cases.



Figure 3. Sample trajectory comparing the four controllers on a test set point

The four controllers were compared for their time of flight, ISE, rise time and settling time averaged across the 50 test cases (Table 1). All controllers perform better than the vanilla LQR (identity Q matrix). LQR combined with RL agent to choose the Q matrix gives the lowest average time of flight, ISE, rise time, and settling time. The RL agent’s flight time is 24% better than the second best method to choose the Q matrix (Bryson’s rule).

Table 1. Comparison of controllers for reaching a set point

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Controller** | **Method of choosing  Q matrix** | **Average Time of flight  (seconds)** | **Average Integral Square Error** | | | | **Average Rise  time (seconds)** | **Average  settling time (seconds)** |
|  |  |  | **x** | **y** | **z** | **psi** |  |  |
| LQR | Identity matrix | 9.83 | 0.37 | 0.35 | 0.38 | 2.44E-07 | 2.45 | 4.38 |
| LQR | Bryson's rule | 6.11 | 0.36 | 0.33 | 0.35 | 8.36E-07 | 1.21 | 2.00 |
| LQR | Reinforcement Learning | 4.64 | 0.15 | 0.14 | 0.11 | 4.08E-05 | 0.52 | 1.71 |
| LQR + PD | Bryson's rule | 9.40 | 0.51 | 0.22 | 0.39 | 1.38E-02 | 1.40 | 3.42 |

* 1. **Test 2: Following a given trajectory**

There are two main differences between the tasks of reaching a set point and following a trajectory. First, after reaching a waypoint, the drone starts moving towards the next waypoint with non-zero linear and angular velocities. Second, the drone does not have to stop at intermediate waypoints and must only come to a halt at the final waypoint. These differences make the problem of following a trajectory worth exploring. The experiments in this section aim to answer the following questions:

1. Is the learning based approach able to follow a desired trajectory?
2. How does performance of learning based approach compare with the following benchmarks:
   1. LQR with identity Q and R matrices.
   2. LQR with Bryson’s rule to tune Q and R matrices.
   3. LQR and PID control with Bryson’s rule to tune Q and R matrices in the LQR.

**Overview of training process**

At the beginning of each episode, a random trajectory with ten waypoints is generated. Each waypoint is sampled using the equations defined below:

Where, and are two consecutive waypoints and and are sampled from a uniform distribution with -1 and 1 as the lower and upper bound, respectively.

The drone starts from an initial state, . In each iteration of the algorithm, the RL agent receives the drone's position relative to the next waypoint and current velocities. For example, if the drone has reached waypoint , the input to the RL agent () will be as follows:

Defining the input in this way allows us to facilitate learning by generalising the goal of the RL agent to a zero vector for all inputs received.

The actor outputs the values of the Q matrix as actions using the state of the drone provided as input. These values are then fed into the LQR, which gives control inputs for the drone.

**Results**

*Training*

With the architecture and reward function described in section 4.3 and the training process described above, the RL agent was trained for 100 episodes. The figure below shows the reward obtained per episode and the average reward obtained till the current episode.

Figure 4. Training progress of the RL agent to follow a trajectory

From the above figure, it can be seen that the agent earns progressively better rewards. As the agent learns, it is able to complete the trajectory more often in the given time limit. During the first few episodes, the agent explores (see section 4.3.2), and after around 15 episodes, it exploits the learned policy and the reward obtained per episode plateaus. This can also be seen in figure 5, in which the number of iterations taken to complete an episode reaches a stable lower value after 15 episodes.



Figure 5. Number of iterations taken by RL agent to complete the trajectory per episode

*Testing*

The RL agent’s ability to follow a trajectory was tested on fifty new random trajectories that were not seen during training. This performance was compared with the three benchmark approaches for selecting the Q matrix in LQR. Figure 6 shows the performance of all four methods for one such test trajectory. The RL agent takes a significantly lower time compared to the other controllers.



Figure 6. Sample trajectory comparing the four controllers on a test trajectory

We calculated the average time of flight and ISE for all four controllers over the fifty test trajectories. Like test 1, the RL agent has the lowest flight time and ISE. Bryson’s rule also gives significantly better results than vanilla LQR, which uses an identity matrix. Combining LQR with PD control decreases the performance, as seen in Table 2 below.

Table. 2 Comparison of controllers for following a trajectory

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Controller** | **Method of**  **choosing Q matrix** | **Average Time of flight (seconds)** | **Average Integral Square Error** | | | |
|  |  |  | **x** | **y** | **z** | **psi** |
| LQR | Identity matrix | 51.64 | 3.60 | 3.06 | 3.36 | 0.000002 |
| LQR | Bryson's rule | 26.98 | 3.40 | 2.73 | 2.93 | 0.000007 |
| LQR | RL | 18.92 | 1.41 | 1.16 | 1.02 | 0.000340 |
| LQR + PD | Bryson's rule | 47.44 | -8.15 | -1.93 | -5.35 | -0.137175 |