# Calculations for Embedded Control Minor Project

## State space equations of the system:

## Linearizing the system:

At the equilibrium point

And substituting

Finding the Jacobian Matrices A and B:

Therefore,

And,

Therefore,

Now, with the below given parameters of the self-balancing bike:

The matrices A and B now become:

Therefore, we have the state space representation of the linearized system in the form:

where,

To find the feedback gain matrix k, methods like pole placement can be used. But (write about disadvantage of pole placement). Therefore, it was decided to design an LQR controller to compute the feedback gains.

But first, it is important to check the controllability of the system by checking the rank of the controllability matrix C.

Using the below MATLAB code,

% system matrices

A = [0, 1, 0;

93.5465, 0, 0;

0, 0, 0];

B = [0;

-338.406;

11515.3];

% controllability

controllability\_matrix = ctrb(A, B);

rank\_of\_controllability = rank(controllability\_matrix);

if rank\_of\_controllability < size(A)

disp('The system is not controllable.');

else

disp('The system is controllable.');

end

The rank of the controllability matrix is found to be 3, which means that our system is fully controllable since this satisfies the requirement of all 3 states of our system to be controllable.

## Designing the LQR Controller:

Given the linearized state space representation:

where:

1. Defining the LQR Cost Function:

The LQR controller will minimize the cost function:

where Q and R are weighting matrices. These matrices determine the trade-off between state deviations and control effort.

1. Choosing Q and R matrices:

Lets start with matrices Q and R being:

1. Continuous-Time Algebraic Riccati Equation (CARE):

To find the gain matrix K, we have to solve the equation CARE given below:

The solution P to this equation will help us compute the gain matrix K:

1. Computing the LQR Gain Matrix K:

Using the below MATLAB code with the ‘lqr’ function:

% the weighting matrices q and r

Q = [1, 0, 0;

0, 1, 0;

0, 0, 1];

R = 1;

% computing the LQR gain matrix K

[K, S, e] = lqr(A, B, Q, R);

% print gain matrix K

disp('The LQR gain matrix K is:');

disp(K);

% print the solution S to the Riccati equation

disp('The solution to the Riccati equation S is:');

disp(S);

% print the closed-loop eigenvalues

disp('The closed-loop eigenvalues are:');

disp(e);

With the lqr function in MATLAB we can also get the solution to the Riccati equation and the eigenvalues of the closed loop system.

With the trial and error method of changing the values of Q and R and implementing the resulting gain matrix K on the self-balancing bike, the optimal gain values for the hardware can be found.

Using these feedback gain values, and the solution of the Riccati equation, the stability of the system is also verified using the Lyapunov stability analysis in the next section.

## Verifying Stability Using the Lyapunov Function:

We will use the following Lyapunov function:

The time derivative of V(x) along the trajectories of the system is:

Using the system dynamics

should be negative definite for the system to have stability.

Therefore, for to be negative definite:

The below given MATLAB code was written for this task:

% matrix of the closed loop system

A\_cl = A - B \* K;

% Lyapunov function matrix P (from the solution to the Riccati equation)

P = S;

Lyap\_matrix = A\_cl' \* P + P \* A\_cl;

% print the Lyapunov matrix

disp('The matrix A\_cl^T \* P + P \* A\_cl is:');

disp(Lyap\_matrix);

% eigenvalues of the Lyapunov matrix (should all have negative real parts)

Lyap\_eigenvalues = eig(Lyap\_matrix);

% print the eigenvalues of the Lyapunov matrix

disp('The eigenvalues of the matrix A\_cl^T \* P + P \* A\_cl are:');

disp(Lyap\_eigenvalues);

% check if all eigenvalues have negative real parts

is\_negative\_definite = all(real(Lyap\_eigenvalues) < 0);

% print whether the Lyapunov matrix is negative definite or not

if is\_negative\_definite

disp('The matrix A\_cl^T \* P + P \* A\_cl is negative definite.');

else

disp('The matrix A\_cl^T \* P + P \* A\_cl is NOT negative definite.');

end

We check if the matrix is negative definite by checking if all the eigenvalues have negative real parts.

By varying Q and R of the LQR controller, different feedback gain matrices were found and tested on the self-balancing bike and their Lyapunov stability was also checked.

Below given are the values of Q and R and their resulting gain matrices K:

(make a table of the values)

The optimal values found to be working good on the self-balancing bike are:

(put the optimal values of Q, R and K here)

## Forward motion, turning and WIFI Control:

Now that the bike can be balanced well on place, the next steps are:

1. To try moving the bike in straight motion and balance it.
2. Make the turning maneuvers (calculate change in theta using the formula of turn radius and put the resulting value of theta in the bias term in the Simulink model)
3. Get the bike working on WIFI connection