**Bryson Rule**

Apply Bryson’s rule to all 12 state variables.

**Code:**

% variable that stores all state values

state\_storage(:,end+1) = x - x\_des;

% check if we have more than 100 values and take rolling window

if size(state\_storage,2)>100

storage\_rolling= state\_storage(:,end-100:end);

else

storage\_rolling = state\_storage;

end

% maximum square of state error.

max\_state\_sq = max(storage\_rolling.^2,[],2);

% only divide if value is not close to 0.

max\_state\_sq(max\_state\_sq<1e-6) = 1;

Q = eye(12,12)./(max\_state\_sq);

**Plot**: Q values as a function of time in the following order:

Calendar

Description automatically generated with medium confidence

Observation: There are huge penalties on terms that allow positions to change: .

Plot: Compare **Bryson’s rule + LQR** vs **LQR** (without applying Bryson’s rule)

A picture containing text, sky, map, orange

Description automatically generated

Observation: Bryson’s rule + LQR does not do well.

**Intuition:** Apply Bryson’s rule only on state variables that we want to control: . This will allow other state variables to vary freely to reach desired .

**Plot**: (next page) Q values as a function of time in the following order:

Table, calendar

Description automatically generated with medium confidence

Plot: Compare **Bryson’s rule + LQR(on x, y, z, phi)** vs **LQR** (without applying Bryson’s rule)

A picture containing diagram

Description automatically generated