Controlling the Compass Gait

Controlling the Compass Gait (Part I)

0.0/8.0 points (ungraded)

In this problem we will stabilize the passive limit cycle we obtained in the previous problem using a time-varying LQR (TVLQR) controller. The first step in order to do this is to modify the standard continuous-time TVLQR approach to work for hybrid systems. The Riccati equation associated with this problem is the so-called "Jump Riccati Equation" (JRE):

$$-\dot{S}(t) = Q - S(t)\,B(t)\,R^{-1}B(t)^T S(t) + S(t)\,A(a) + A(t)^T S(t) \quad (1)$$
$$S(t^-) = A_d^T S(t^+)\,A_d. \quad (2)$$

Here, equation (1) is the usual evolution of the cost (same as the continuous time case), which we solve backwards in time with a final-value constraint $S(t_f) = Q_f$. The second equation propagates the cost backwards through a hybrid event ("jump"). Here A_d is the linearization of the discrete transition dynamics (in our case, it is the linearization of the collision dynamics for the compass gait).

The second modification we need to the standard continuous-time TVLQR is to make its solution periodic. In other words, we want the cost-to-go at $t = 0$ to be the same as the cost-to-go at the final time $t = t_f$ (since the limit cycle is periodic). It is well-known (but not trivial to prove!) that repeatedly solving the equations (1) and (2) backwards in time will result in convergence to a periodic $S(t)$. Hence, we solve (1) backwards in time for the interval $[t^+, t_f]$ with a final-value constraint $S(t_f) = Q_f$. Then, we solve (2) using the $S(t^+)$ obtained from (1). We then solve (1) again with the final value constraint $S(t_f) = S(t^-)$, and so on until convergence.

We have provided stub code here that solves the continuous time equation (1) using Drake. Your task is to fill in the section corresponding to the discrete jump equation (2), and to check for convergence of $S(t)$ to a periodic solution. It might be helpful to take a look at the function **collisionDynamics** in CompassGaitPlant.m. The inputs to this function are the hybrid mode ("1" in our case), time, pre-impact state (a 4×1 vector) and control input. The first output is the post-impact state. The columns of the last output (dxp) are the gradients of the post-impact state with respect to the different inputs of the function. So, the columns of interest to us are the columns 3 through 6 (since these are the gradients with respect to the pre-impact state).

Note: This problem relies on your answer(s) from the previous problem. To be safe, make sure to use all the digits you comptued in Matlab in the previous part. Also set your tolerances much tighter than the ones we list in the problems (since there is almost no computational cost to doing this).

(a) What is the value of Q_f when the solution has converged (the variable "Qf_converged" in the code)? **Our tolerance for this answer is 0.01.**

```
Qf_converged = ;

```

Unanswered

Run Code

Submit You have used 0 of 3 attempts

Controlling the Compass Gait (Part II)

0.0/7.0 points (ungraded)

(b) Next, implement a function similar to the function strideFunction in the previous problem in order to simulate the closed-loop system for one step. The code we have provided you in stabilizeCompassGait.m shows you how to simulate the closed loop system ("sysClosedLoop in the code"). Use this to compute the Floquet multipliers for the closed-loop system just as you did in the previous problem for the passive system. Type in your answer below as a 4×1 vector of complex numbers. Don't worry about ordering the multipliers in any particular order. **Our tolerance for this answer is 0.05 (in terms of the magnitude of each element in the vector).** (Compare the largest Floquet multiplier with the largest multiplier you obtained for the passive system in part (d) of the previous problem. Is it smaller or bigger in magnitude? Is this what you expected?)

```
f = ;

```

Unanswered

Run Code

Submit You have used 0 of 3 attempts