Scaling Chains of Integrators

Rohit Sheth, Nishan Srishankar, Joseph McMahon, and Srishti Srivastava

***Abstract*—Recent advances in computing power have enabled greater levels of simulation, and more detailed modeling. The Chains of Integrators problem is to control a point mass robot in 2D or 3D with only one control input at the nth order derivative (typically, this can be done up to 5th or 6th order derivatives). For example, in a 2nd order system, acceleration is controlled, in a 3rd order system, jerk is controlled, and in a 4th order system, snap is controlled.**

The problem specifically states that the system should reliably visit one or more goal regions, while avoiding obstacles, and then return to the origin.

This work shows a simple solution using the provided LQR controller to track waypoints along a collision free path generated by the A* algorithm. Future implementations for the ICRA FMRB contest are also discussed.

I. Introduction

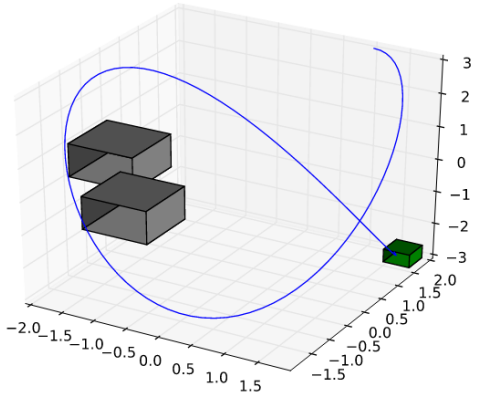


Fig. 1: Visualization of a path in 3D state space with obstacles.

Optimal control has been successfully simulated for low order dimensions and control derivatives. High-dimensional and high-order system simulation allows for high accuracy and robust control in trajectory tracking. The goal of this project is to simulate the most generalized LQR (or LQG) controller possible within the given challenge time (deadline submission for simulation is May 8th). This involves generating a trajectory (shown in Figure 1), motion planning, and optimal control.

$$q(0) = init \wedge \Box(q(t) \notin Obs) \wedge \bigwedge_i \Box\Diamond goal_i \tag{1}$$

In constraint equation (2), a point mass attempts to maneuver from an initial state and reach a goal within a given time period while avoiding obstacles.

$$q(0) = init \wedge \Box(q(t) \notin Obs) \wedge \bigwedge_i (counter_i \not\leqslant T_i) \bigcup goal_i \tag{2}$$

where $Obs \subset \mathbb{R}^n$; $init \in \mathbb{R}^n$.

A cost-value function will be used evaluate initial state, intermediate trajectory states and all goal regions in conjunction with the temporal logic constraints in equations 1 and 2 above. A motion planning algorithm, A*, is implemented to satisfy the temporal logic equation to avoid obstacles while visiting all goal states. The path generated is then sampled down to 10 points, and these points are fed into the LQR controller which finds the optimal way to connect them.

A. Problem Statement

A simple, graphical way to understand the problem is shown in the generated simulations in Figure 2. The point mass starts at a random position in each case (sometimes inside of an obstacle), and it must visit all the randomly generated goals while avoiding all of the randomly generated obstacles.

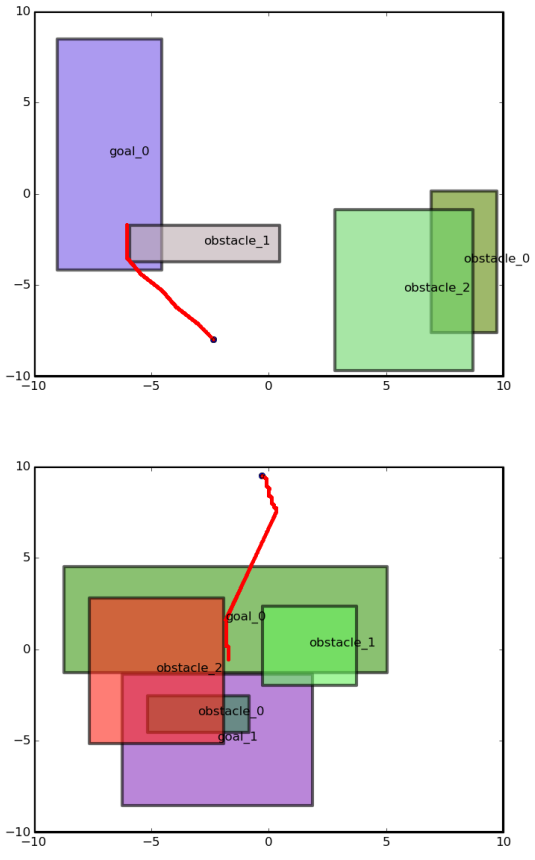


Fig. 2: LQR tracking A* generated paths.

The goal is for the point mass to move through the goal regions without hitting obstacles. This requires trajectory gen-

eration, motion planning, and controls. The dynamics of the system are purely linear, and require little analysis.

The temporal logic equation can be depicted as an automaton for $i = 1$ as shown in Figure 3.

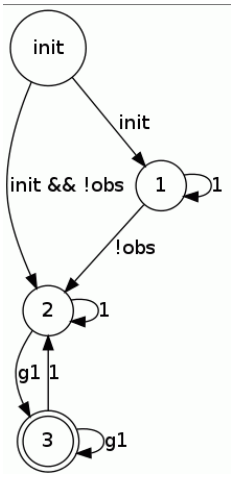


Fig. 3: Buchi Automaton For given LTL equation for one goal and one obstacle.

B. Contest Criteria

1) Decide realizability. Wrong answer is worse than no answer.
2) Time between receiving problem instance and declaring answer to #1.
3) Any violation of task formula is worse than not trying to answer the problem instance.
4) Provided all of the above are successfully met (and provided that the problem instance is realizable), synthesis times are recorded for increasing:
   a) Number of integrators
   b) Number of state dimensions
   c) Number of goal regions
   d) Number of obstacle regions.
5) Let the cycle-time be the time required to visit every goal region at least once. The minimum and maximum cycle-times in a trial are recorded.

Noise and disturbances are not yet implemented, and given time constraints before ICRA 2016, solutions should first be directed to the noise and disturbance free cases.

II. Dynamics

The dynamics of the system are relatively simple as the problem requires controlling a point mass. A two dimensional system (double integrator) can be expressed as a linear system of equations given below:

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u + \xi$$
$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} + \eta$$

where ξ and η are either bounded disturbances (non-deterministic) or stochastic processes.

The system given by the differential equation $D^m q = u$ is called a chain of integrators because it can rewritten as a series of linear control equations as shown below.

$$\begin{aligned} \dot{x_1} &= x_2 \\ \dot{x_2} &= x_3 \\ &\dots \\ \dot{x}_{m-1} &= x_m \\ \dot{x}_m &= u \\ y &= x_1 \end{aligned}$$

At first, we use several assumptions until simpler simulations work, after which the problem will be generalized further. Namely, the A* algorithm is used to take random collision-free steps and evaluate their proximity to the next goal region. Then once a collision-free path is generated, it is sampled (say 10 points along it) and these waypoints are passed to the LQR controller.

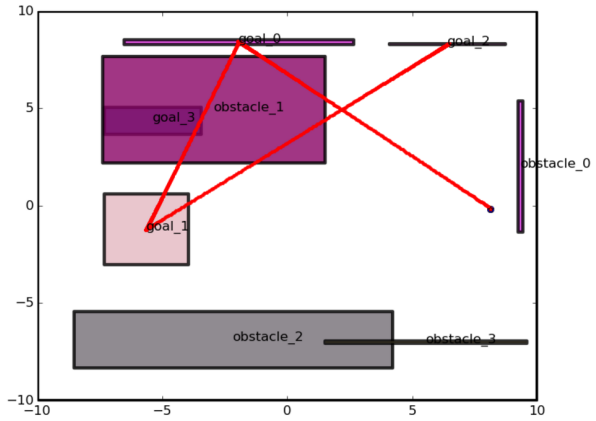


Fig. 4: Simulation of the LQR system provided by the ICRA FMRB team. It does not have any motion planning.

This method is very similiar to the system provided by the ICRA FMRB team. That system does not have any motion planning, and passes the center of goals directly to the LQR as shown in Figure X. The path goes directly though an obstacle.

III. Controls

The FMRB contest is a controller contest. No matter the solution provided, it must implement a controller of some sort. The contest criteria are focused on scalability – not optimality. Scalable linear matrices and their forumlation with the LQR controller are presented.

A. Formulating Scalability Equations

The scalability of the system is the primary contest criteria. The entire system is easily linearized because it is an arbitrary sized chain of integrators. This means the system can be conveniently modeled using linearly independent matrices and state vectors.

The linear system is expressed as below.

$$\dot{x} = Ax + Bu$$

$$y = Cx + Du$$

If we have n dimensions and we define each dimension by $\begin{bmatrix} x & y & z & \theta & ... & \gamma \end{bmatrix}$.

$$x_{mn} = \begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \vdots \\ \dot{y_1} \\ \dot{y_2} \\ \vdots \\ \dot{y_m} \\ \vdots \\ \dot{\gamma_m} \end{bmatrix} \tag{3}$$

$$A_m = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & 0 & 1 \\ 0 & 0 & \dots & 0 & 0 & 0 \end{bmatrix} \tag{4}$$

$$B_m = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \tag{5}$$

$$A_{mn} = \begin{bmatrix} [A_m] & 0 & \dots & 0 \\ 0 & [A_m] & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & & 0 & [A_m] \end{bmatrix} \tag{6}$$

$$B_{mn} = \begin{bmatrix} B_m \\ B_m \\ \vdots \\ \vdots \\ B_m \end{bmatrix} \tag{7}$$

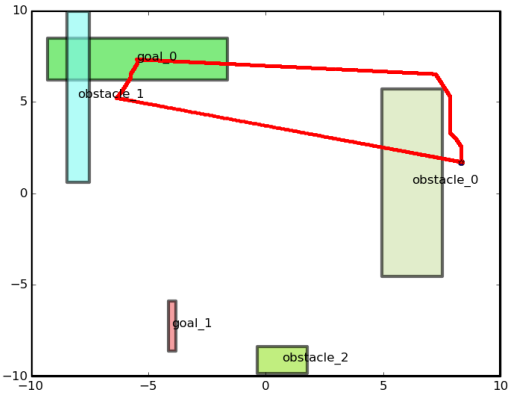
$$C_{mn} = (B_{mn})^T \tag{8}$$

$$D_{mn} = 0 \tag{9}$$

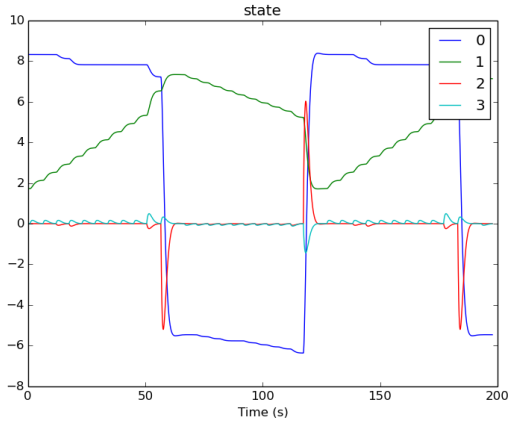
B. Linear Quadratic Regulator

LQR (Linear-Quadratic Regulator) is the solution to a set a linear differential equations, such as those given in the problem. The cost of the dynamics is a quadratic function as shown in Equation 10. The cost function is a solution to the Linear-quadratic problem where Q, Q_f and R are the state cost, final state costs, and input cost matrices. This also extends to the LQG problem, which is LQR with added white noise.

$$J = \int_0^T (x(\tau)^T Q x(\tau) + u(\tau)^T R u(\tau)) d\tau + x(T)^T Q_f x(T) \tag{10}$$



(a) Trajectory of LQR tracking A* path.



(b) Evolution of State Vector while LQR tracks A* path.

Fig. 5: Simulation of LQR tracking A* generated path. It successfully avoids one obstacle and reaches one goal region. After that the code still needs to be improved.

IV. Motion Planning

The following assumptions regarding obstacles will be made in motion planning.

- Static Obstacles.
- Convex Obstacles.

As a minimum goal, a manually generated trajectory satisfying the temporal logic conditions is fed to the controller.

As a reach goal, a visibility graph based method is used to divide the entire space in discrete triangles. A set of trajectories are generated by connecting midpoints of adjacent triangles. Trajectories which do not satisfy temporal logic condition are eliminated. This leaves a set of trajectories which visit all goal regions while avoiding the obstacles. Shortest distance trajectory is then selected for planning the motion.

Furthermore, A* can be implemented on mid-points of sections generated by the visibility graph to reduce execution time and high number os explored states to obtain a quick path.

A. A*

A* is a best-first-search algorithm which can be solve problems by searching in a pre-determined 'greedy' manner to find an apparent quick path to a goal. Paths is estimated by a cost function which is the sum of cost-from start and cost-to-go to goal. Inflating the cost-to-go will determine the greedy approach of A*. If the cost-to-go is not admissible, the algorithm may not find the shortest path.

```
while Queue.empty == false do
    read current-state;
    if current-state is goal then
        break;
    end
    for next-state in neighbours of current-state do
        new-cost = cost-from(start) + cost-to-go(goal);
        if next-state not unique or new-cost less than
          old-cost then
            old-cost = new-cost;
            priority = new-cost + heuristic(new-state to
              goal);
        end
        if next-state not in collision then
            Queue.put(next-state, priority);
        end
        parent(next-state) = current-state;
    end
end
```

Algorithm 1: A* Algorithm

V. Progress

The competition dry run simulation is on May 1st. The real contest simulations take place May 8th and May 15-16th. The Gaussian error and disturbances, and optimal solutions are not the primary concern. Scaling a solution that meets all of the specifications is the goal, and evaluating realizablity is important because a partial or invalid solution is much worse than no solution at all.

The benchmark software comes with a generator which generates random goals and obstacles every time it is run. This provides the world for the point robot to move in, which itself can be spawned anywhere in the world, including inside obstacles. Similarly, the goals can also be encased in obstacles, in which case our program should identify that it is an unsolvable problem.

Once the world and the robot are successfully generated, the A* algorithm generates a path from the initial point visiting all goal regions while avoiding obstacles as shown in Figure 5. This path is sampled to give 10 waypoints along the collision free path that are fed to the LQR, which in turn tracks them. This method is almost working per the contest's scaling specification.

References

[1] Smith, S. L. et al. "Optimal Path Planning For Surveillance With Temporal-Logic Constraints". The International Journal of Robotics Research 30.14 (2011): 1695-1708. Web.
[2] Murray, Richard. Optimization-Based Control. 2nd ed. California Institute of Technology, 2010. Print.
[3] Rantzer, Anders. "Dynamic Programming Via Convex Optimization". (1999): n. pag. Print.
[4] Hespanha, Joao. Linear Systems Theory. Princeton: Princeton University Press, 2009. Print.
[5] Fainekos, Georgios E. et al. "Temporal Logic Motion Planning For Dynamic Robots". Automatica 45.2 (2009): 343-352. Web.