Nonholonomic Control with Turtlebots

Jonathan Lee[*], Ginny Xiao[†]

April 8, 2019

1 Methods

1.1 Bang-Bang Control

We implemented the Bang-Bang controller by cycling between controlling x, ϕ, θ, and y, in that order. For each state variable, we checked whether the true value was within a tolerance of the desired value (these tolerances are $0.05\,\text{m}, 0.05\,\text{m}, 0.1\,\text{rad}, 0.1\,\text{rad}$, respectively). If this value was not within a tolerance, we applied proportional closed-loop control until the variable was sufficiently close to the desired value. This control was simple for controlling x and ϕ, with gains of 0.6 and 0.5, respectively, since these variables could be manipulated directly. For controlling θ, we applied a turning motion with gain 0.6. For controlling y, we applied a strafing ("parallel parking") motion with gain 2.5, where each motion took 0.8 s.

1.2 Sinusoidal Steering

To implement sinusoidal steering, we first steered x and ϕ, the Turtlebot's forward displacement and steering angle of the forward wheels, respectively. Since the robot has direct control over these state variables, this was easy to do by applying a constant velocity.

To steer $\alpha = \sin\theta$ when the desired bearing $|\theta_d| < \pi/2$, we modeled the inputs u_1, u_2 to x, ϕ as sinusoids

$$v_1 = u_1 \cos\theta = a_1 \cos(\omega t) \tag{1}$$

$$v_2 = u_2 = a_2 \sin(\omega t) \tag{2}$$

where Δt is the duration of the sinusoidal maneuver and $\omega = 2\pi/\Delta t$ is its frequency. Setting $a_2 = \max\{1, \omega \min\{|\phi_d - \phi_{max}|, |\phi_d + \phi_{max}|\}\}$ arbitrarily, where ϕ_d and $\phi_{max} = 0.3$ are the desired and maximum steering angle, respectively, we solved for a_1 by numerically integrating:

$$\beta_1 = \frac{\omega}{\pi} \int_0^{\Delta t} f\left(\frac{a_2}{\omega}\sin(\omega t) + \phi(0)\right) \sin(\omega t)\,\mathrm{d}t \quad f(\phi) = \frac{1}{\ell}\tan\phi \tag{3}$$

where ℓ is the length of the Turtlebot. Then, the a_1 that yields a desired change in α (and θ) is $a_1 = (\omega\Delta\alpha)/(\pi\beta_1)$.

For $|\theta_d| \geq \pi/2$ (that is, where the robot's bearing is parallel to the positive or negative y-axis at some point), there is a singularity at $\theta = \pi/2$, so we ignore the canonical model and use a direct relationship with θ instead:

$$\dot{\theta} = f(\phi)\, u_1 \tag{4}$$

[*] jonathan-lee@berkeley.edu, SID 3031879358
[†] ginnyxiao@berkeley.edu, SID 3034306890

We solve for a_1, a_2, given $\Delta\theta$, in the same way as for $\Delta\alpha$, but now no longer need to compute $u_1 = v_1/\cos\theta$, which is undefined at $|\theta| = \pi/2$. Instead, $v_1 = u_1$ directly.

To steer y, we use sinusoidal inputs again:

$$v_1 = u_1 \cos\theta = a_1 \sin(\omega t) \tag{5}$$

$$v_2 = u_2 = a_2 \cos(2\omega t) \tag{6}$$

Setting $a_2 = 0.26$ arbitrarily to saturate ϕ (and create the largest displacements in y possible), we compute

$$\beta_1 = \frac{\omega}{\pi} \int_0^{\Delta t} g\left(\int_0^t f\left(\frac{a_2}{2\omega} \sin(2\omega\tau) \right) a_1 \sin(\omega\tau)\, \mathrm{d}\tau \right) \sin(\omega t)\, \mathrm{d}t \quad g(\alpha) = \frac{\alpha}{\sqrt{1-\alpha^2}} \tag{7}$$

for various guesses of a_1. Then, the resulting displacement is $\Delta y = \pi a_1 \beta_1 / \omega$.

To guess a_1, we use a binary search:

(a) Initializing $a_1^{min} = 0$ and $a_1^{max} = 20$, we guess that $a_1 = \left(a_1^{min} + a_1^{max}\right)/2$.

(b) Using equation 7, we compute Δy.

(c) If Δy is within 0.001 m of the desired Δy_d, then output a_1.

(d) If $\Delta y > \Delta y_d$, then we have an overestimate and set $a_1^{min} = a_1$. Otherwise, we have an underestimate and set $a_1^{max} = a_1$. In both cases, we return to step (2).

If, at any point, $a_1^{min} > a_1^{max}$, then there is no feasible a_1, which we consider a failure.

Because the steering angle ϕ is limited to approximately $(-0.3\,\mathrm{rad}, 0.3\,\mathrm{rad})$, the displacement the Turtlebot can drive in the y-direction is also limited. Therefore, we segmented translations in y into separate translations of, at most, 0.25 m. Similarly, we segmented rotations in θ into separate rotations of, at most, $(\pi/3)$ rad.

2 Results

A video of our implementation is available at: `https://youtu.be/r6tmWgoy7x0`. The code for our implementation is at this GitHub repository:

`https://github.com/jonathan-j-lee/eecs-106b/tree/master/lab-03-steering`

Figures 1, 3, 5, 7, and 9 show the true turtlebot (x, y, ϕ, θ) states reached versus the desired states with the direct Lie algebra controller. Figure 2, 4, 6, 8, and 10 show the desired and true (x, y) paths of the turtlebot.

For all the tasks that use proportional Bang-Bang control, the turtlebot ultimately reaches the desired state with a small error bounded in range $[-0.05, 0.05]$, but in different amounts of time. Sometimes, it takes more than 50 seconds to converge, as shown in figures 7 and 3, and the 2D path plots (figures 8 and 4) show that the turtlebot keeps moving back and forth. While the results justify the robustness of closed-loop Bang-Bang Control, the effectiveness is not very good due to the cost of overshoot and an oscillating steady-state condition.

One difficulty we encountered with the open-loop sinusoidal trajectories was that the Turtlebot seemed to travel less than expected in the x-direction. To solve this, we significantly reduced the speed of the robot by increasing Δt to between 6 s and 14 s. We also multiplied the forward velocity control by a scaling factor between 1.25 and 1.75 to compensate for this weaker drive, which could result in accumulated error.

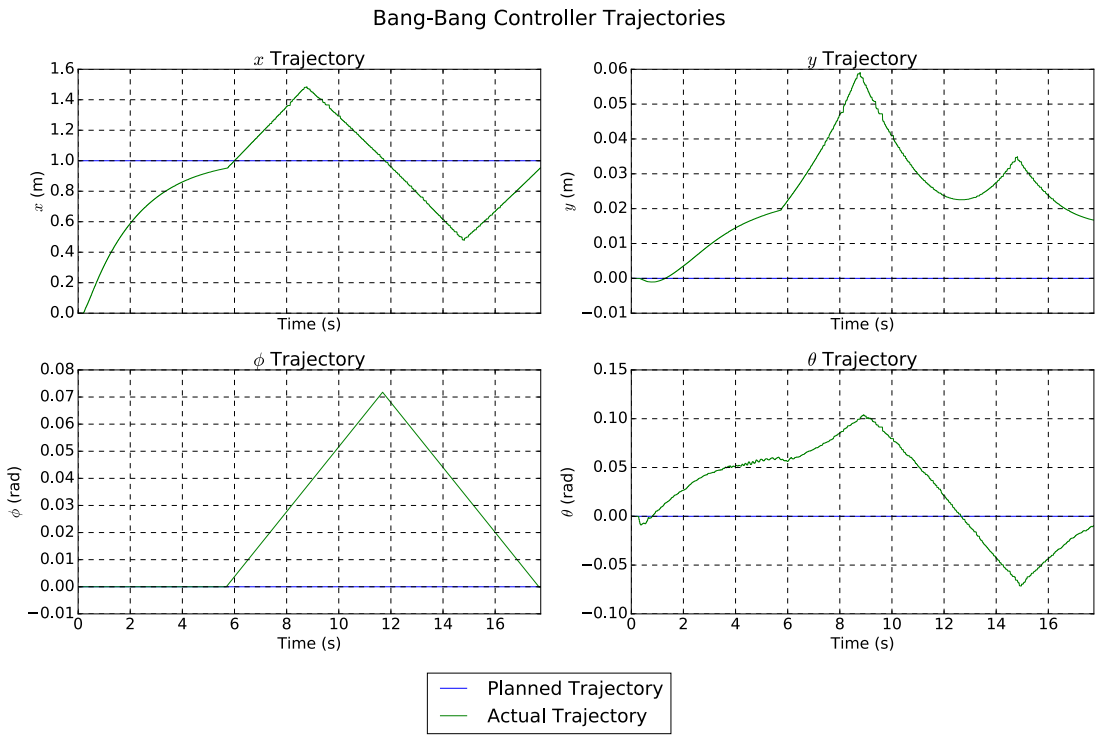

Figure 1: Trajectory with Bang-Bang Control and Desired Final State $(x, y, \phi, \theta) = (1, 0, 0, 0)$

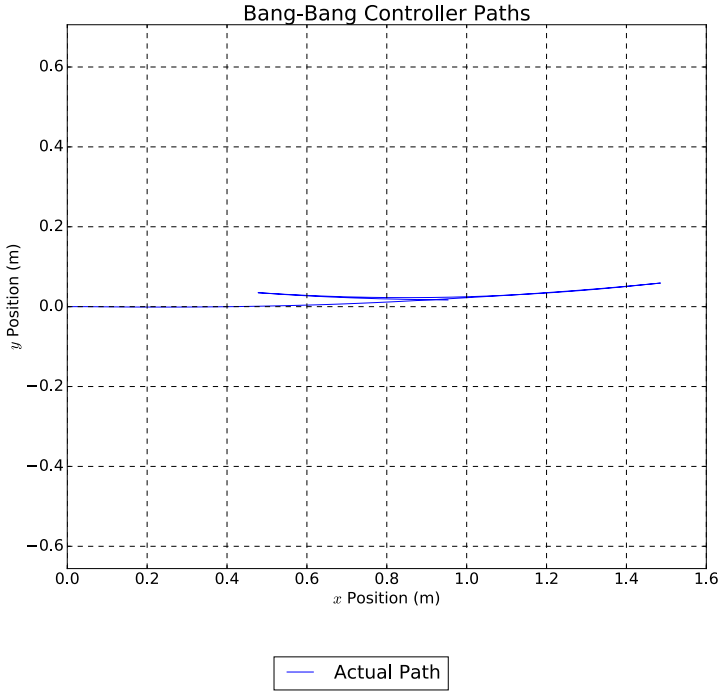

Figure 2: 2D Path with Bang-Bang Control and Desired Final State $(x, y, \phi, \theta) = (1, 0, 0, 0)$

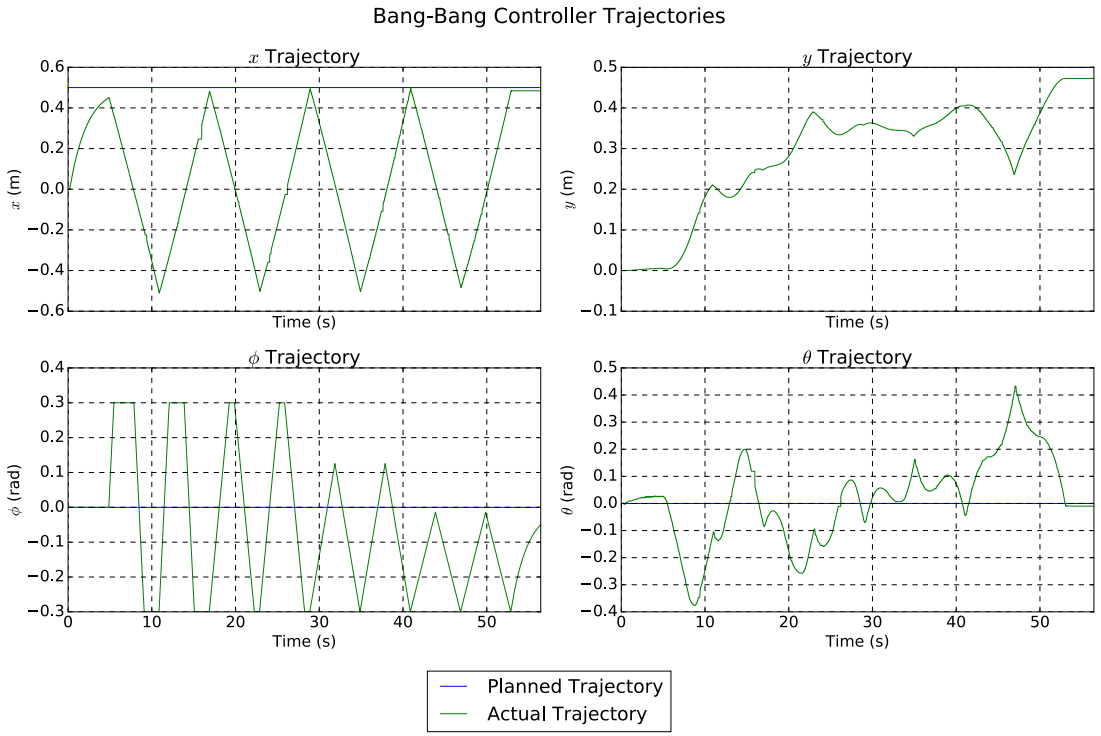

Figure 3: Trajectory with Bang-Bang Control and Desired Final State $(x, y, \phi, \theta) = (0.5, 0.5, 0, 0)$

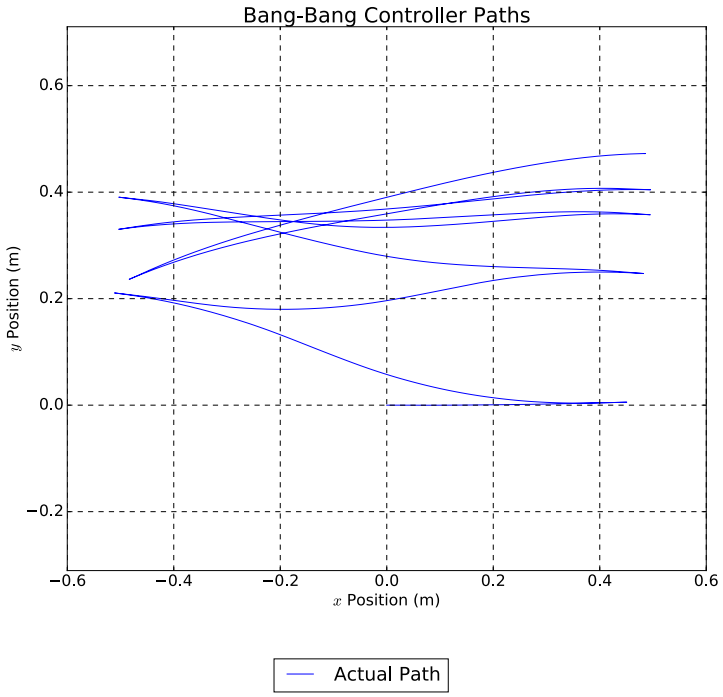

Figure 4: 2D Path with Bang-Bang Control and Desired Final State $(x, y, \phi, \theta) = (0.5, 0.5, 0, 0)$

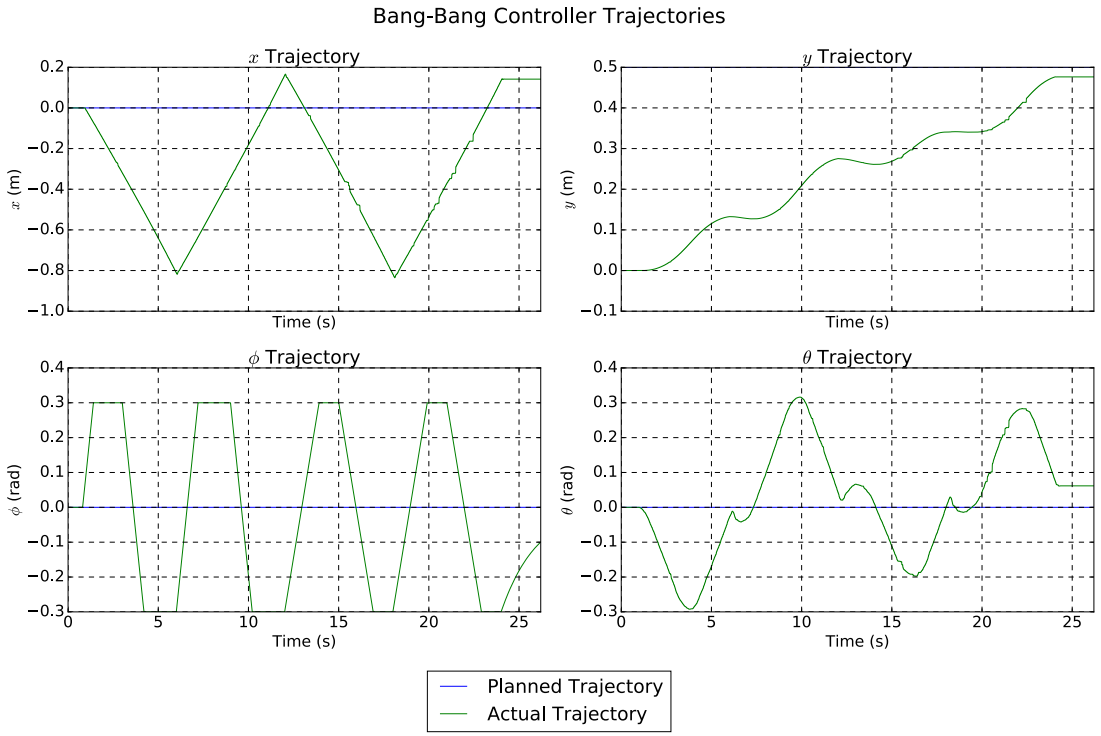

Figure 5: Trajectory with Bang-Bang Control and Desired Final State $(x, y, \phi, \theta) = (0, 0.5, 0, 0)$

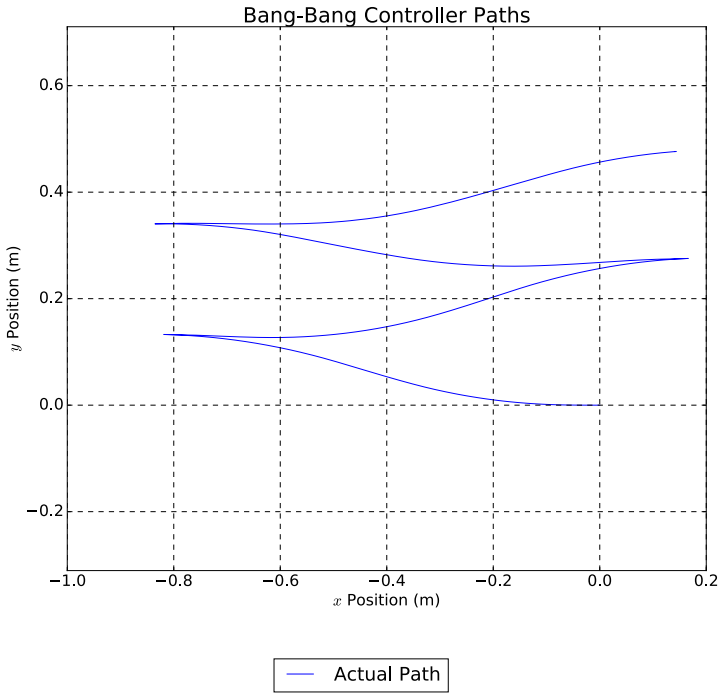

Figure 6: 2D Path with Bang-Bang Control and Desired Final State $(x, y, \phi, \theta) = (0, 0.5, 0, 0)$

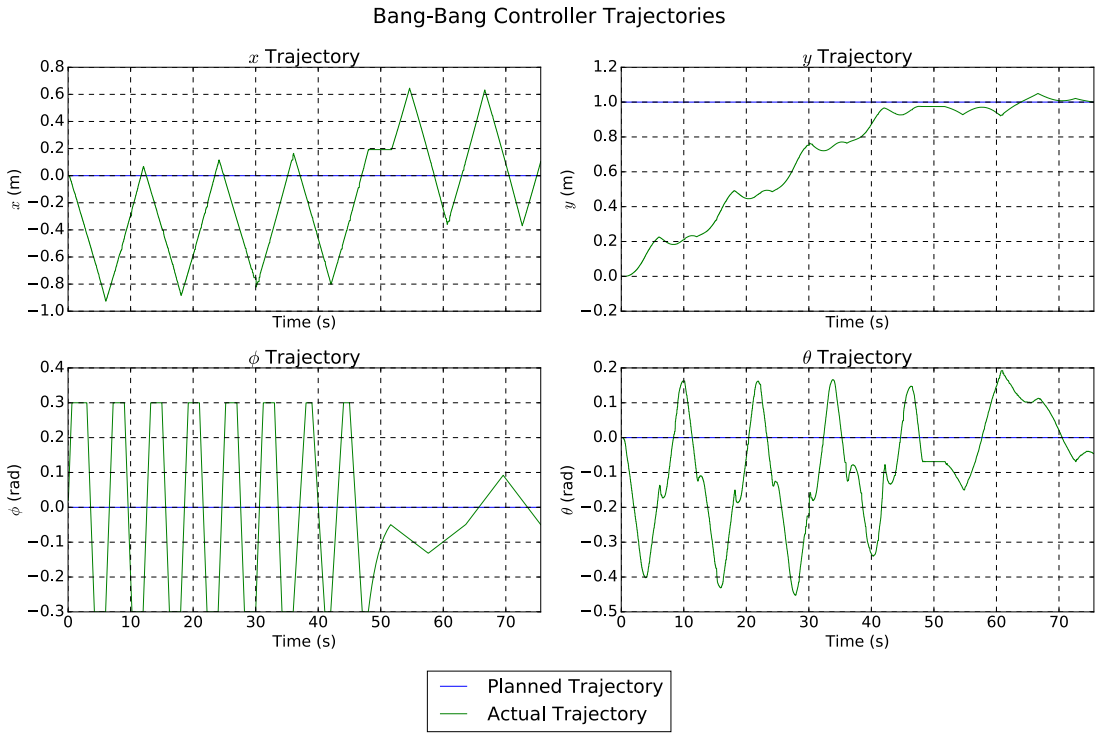

Figure 7: Trajectory with Bang-Bang Control and Desired Final State $(x, y, \phi, \theta) = (0, 1, 0, 0)$

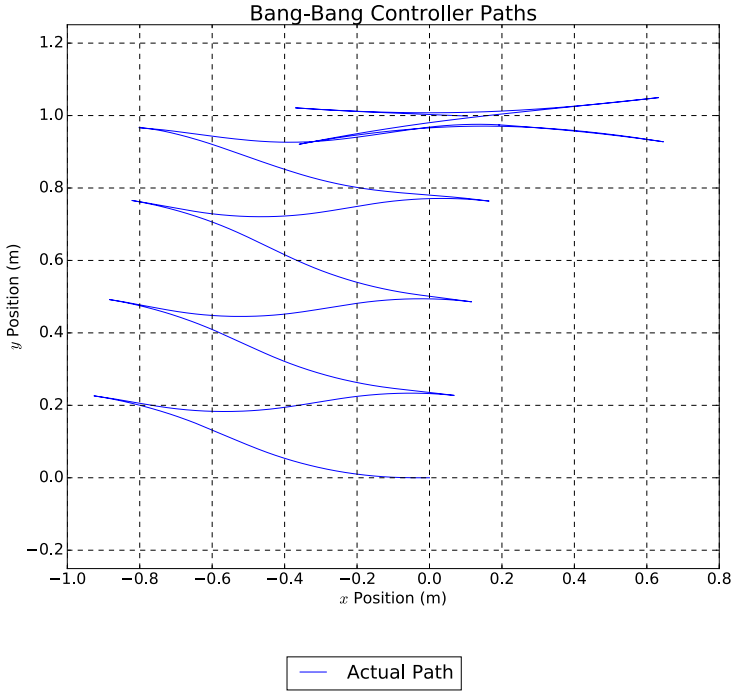

Figure 8: 2D Path with Bang-Bang Control and Desired Final State $(x, y, \phi, \theta) = (0, 1, 0, 0)$

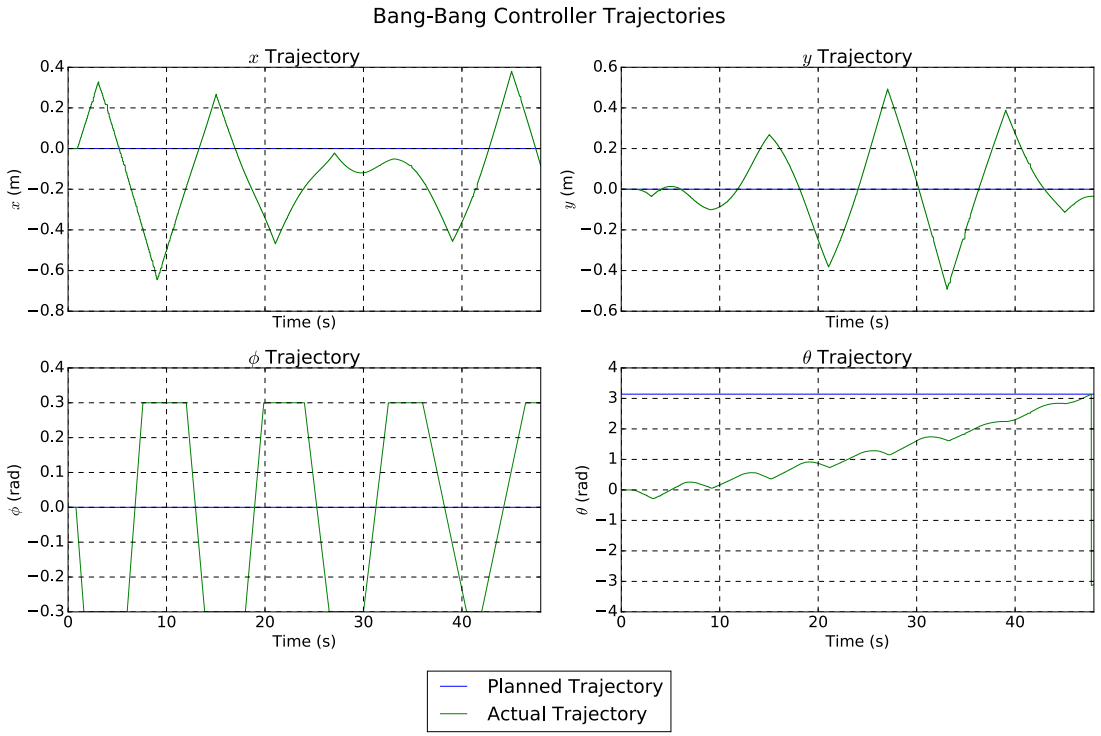

Figure 9: Trajectory with Bang-Bang Control and Desired Final State $(x, y, \phi, \theta) = (0, 0, 0, \pi)$

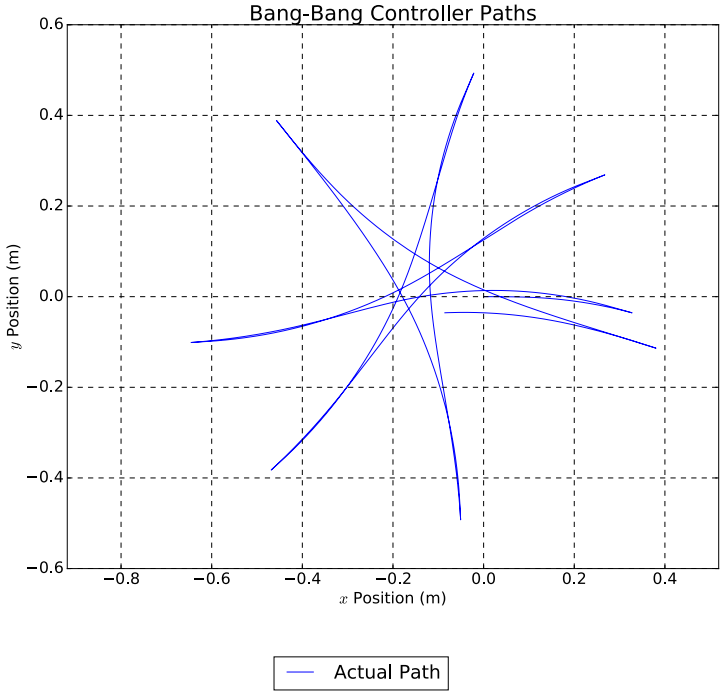

Figure 10: 2D Path with Bang-Bang Control and Desired Final State $(x, y, \phi, \theta) = (0, 0, 0, \pi)$

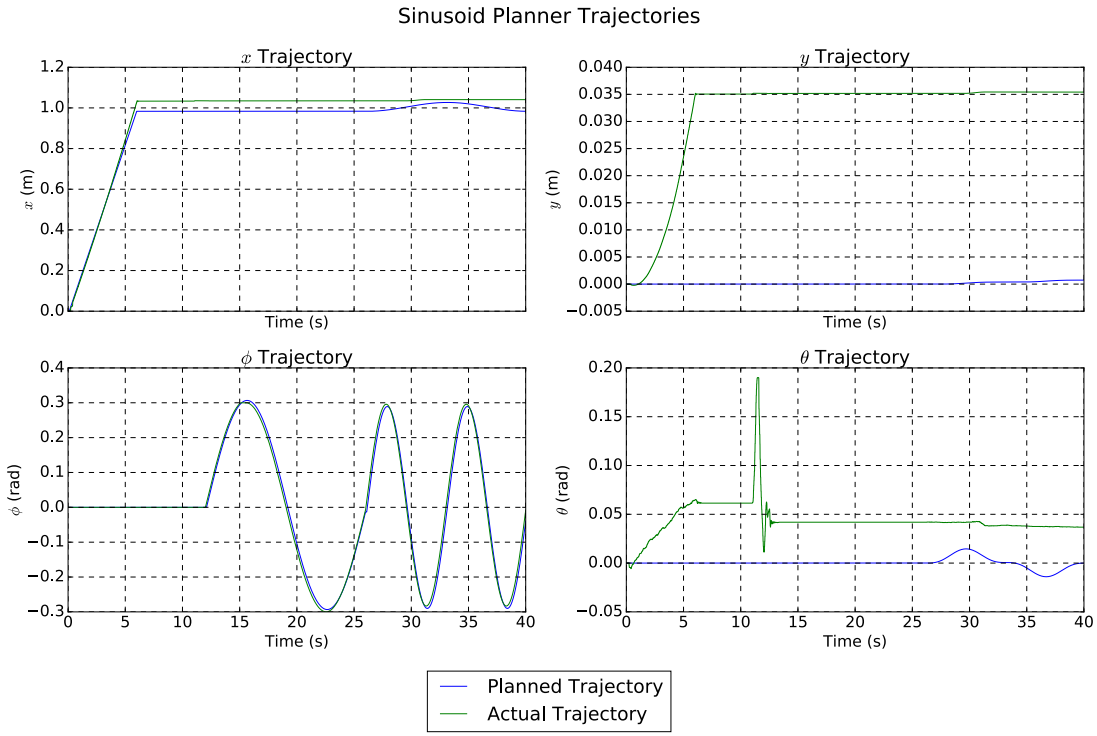

Figure 11: Trajectory with Sinusoidal Steering and Desired Final State $(x, y, \phi, \theta) = (1, 0, 0, 0)$

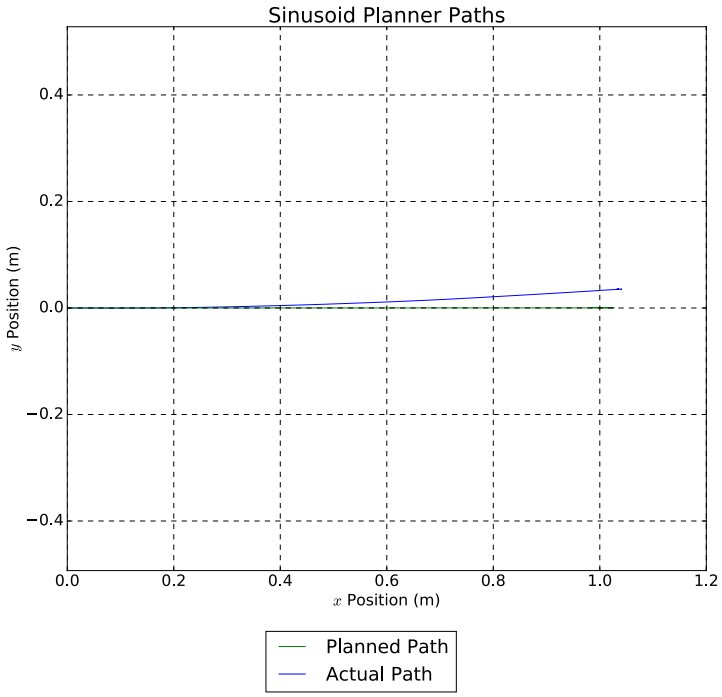

Figure 12: 2D Path with Sinusoidal Steering and Desired Final State $(x, y, \phi, \theta) = (1, 0, 0, 0)$

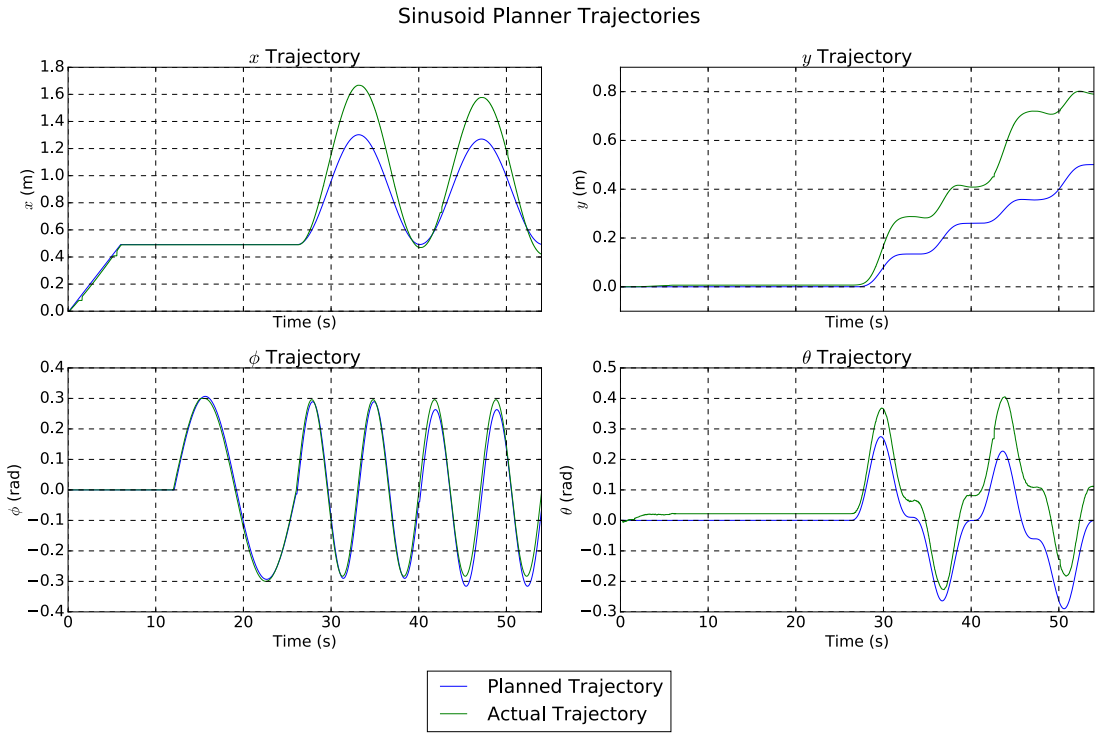

Figure 13: Trajectory with Sinusoidal Steering and Desired Final State $(x, y, \phi, \theta) = (0.5, 0.5, 0, 0)$

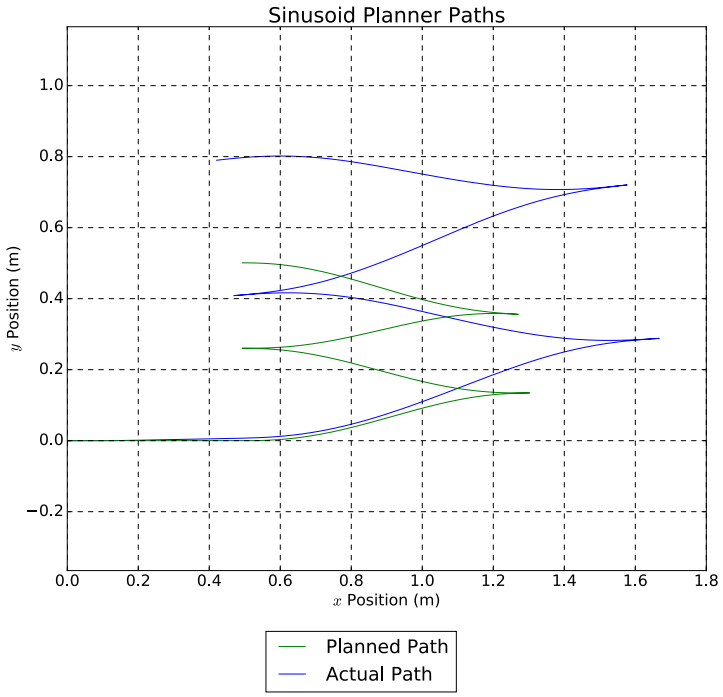

Figure 14: 2D Path with Sinusoidal Steering and Desired Final State $(x, y, \phi, \theta) = (0.5, 0.5, 0, 0)$

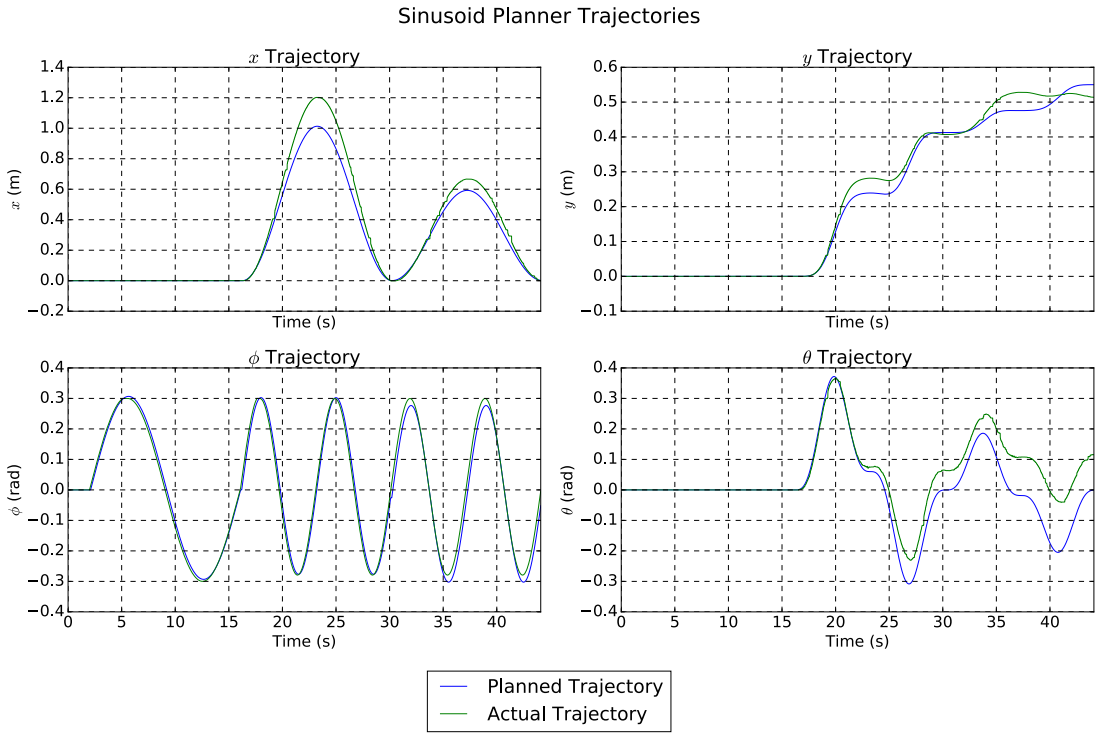

Figure 15: Trajectory with Sinusoidal Steering and Desired Final State $(x, y, \phi, \theta) = (0, 0.5, 0, 0)$

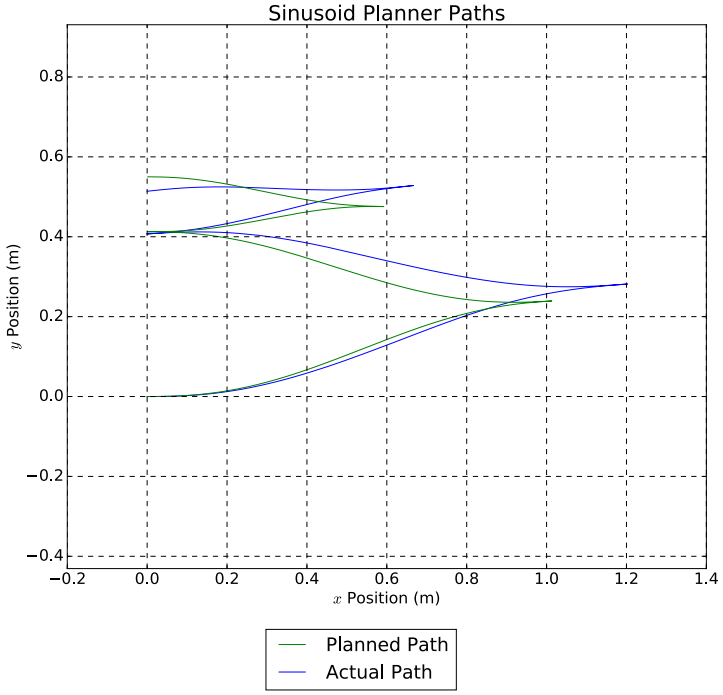

Figure 16: 2D Path with Sinusoidal Steering and Desired Final State $(x, y, \phi, \theta) = (0, 0.5, 0, 0)$

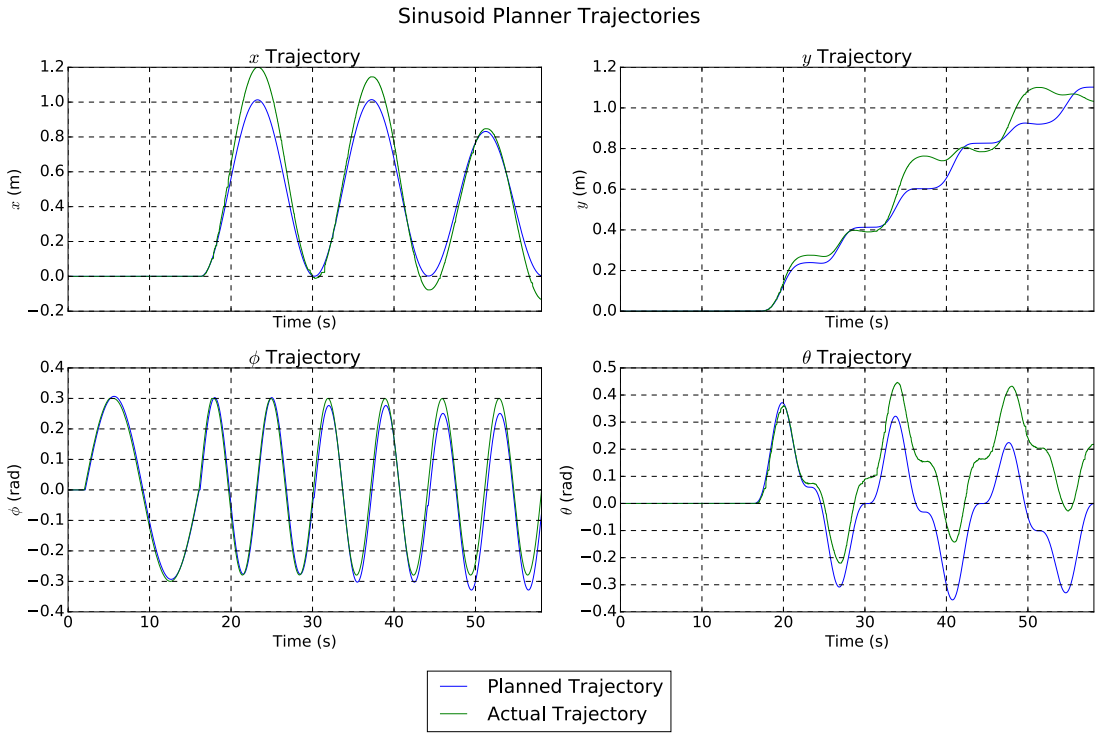

Figure 17: Trajectory with Sinusoidal Steering and Desired Final State $(x, y, \phi, \theta) = (0, 1, 0, 0)$

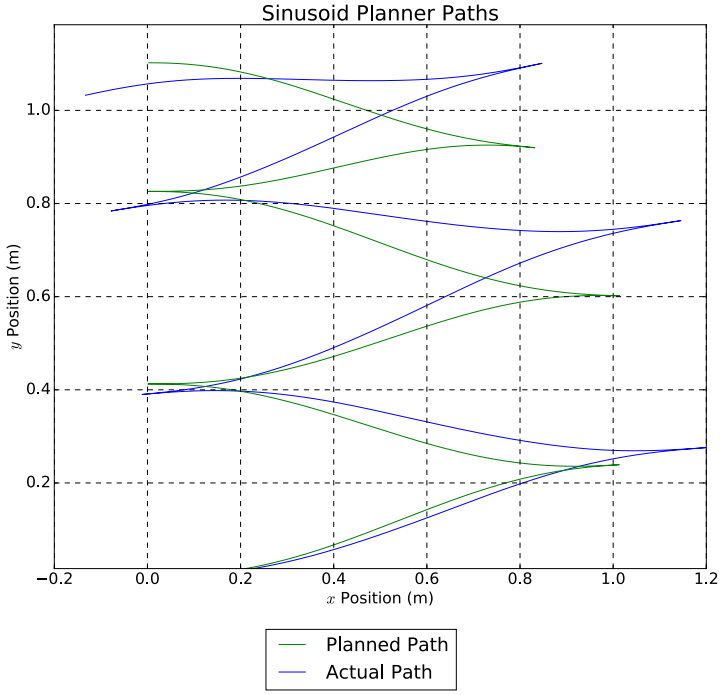

Figure 18: 2D Path with Sinusoidal Steering and Desired Final State $(x, y, \phi, \theta) = (0, 1, 0, 0)$

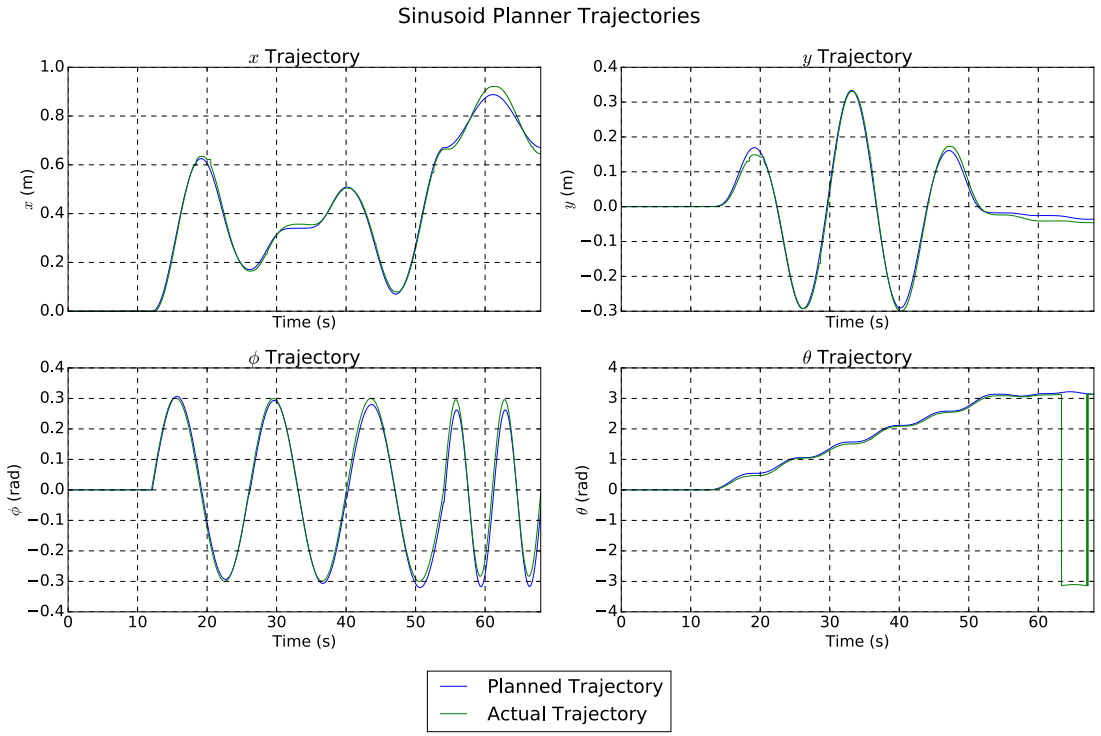

Figure 19: Trajectory with Sinusoidal Steering and Desired Final State $(x, y, \phi, \theta) = (0, 0, 0, \pi)$

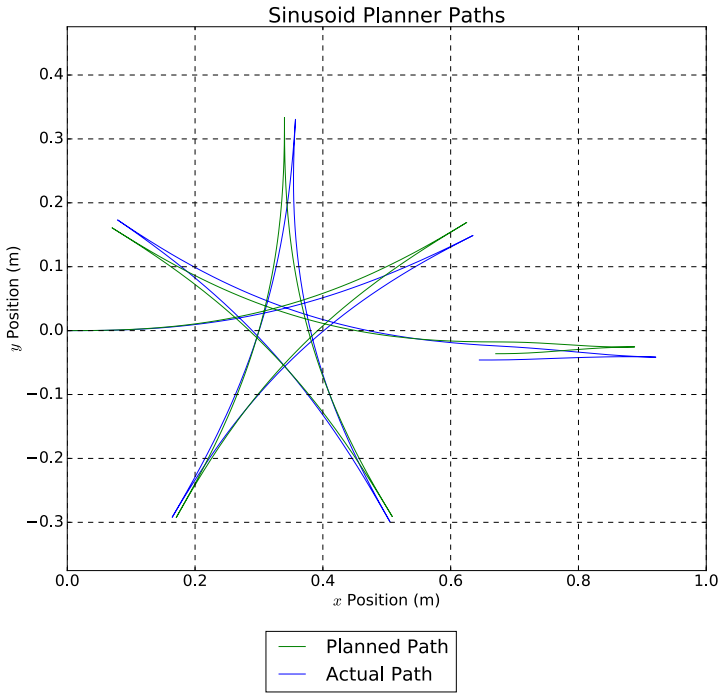

Figure 20: 2D Path with Sinusoidal Steering and Desired Final State $(x, y, \phi, \theta) = (0, 0, 0, \pi)$

3 Discussion

Upon analysis of the two systems, it can be concluded that both the proportional Bang-Bang controller and the Sinusoidal Steering enable the robot to perform the task successfully. With proportional Bang-Bang control the system is able to reach its desired state with a smaller error, but can take a very long time to converge. Also, the trade-off between and steady-state error and the percent overshoot due to the proportional gain tuning is inevitable. With Sinusoidal Steering the system is able to reach the desired state efficiently and smoothly. In this lab, however, the open-loop control scheme for Sinusoidal Steering results in larger errors.

4 Paper Summaries

Motion Planning for a Knife-Edge Moving on the Surface of a Torus [2] studies the motion-planning problem for a knife-edge moving on the surface of a torus. This paper derives a kinematics formulation for a knife-edge moving on an arbitrary smooth surface using global coordinates to avoid singularities. This work builds on Sastry's *Steering on Sinusoids* formulation of nonholonomic systems by developing a novel motion planning algorithm based on the geometry of a torus.

SLAP: Simultaneous Localization and Planning Under Uncertainty via Dynamic Replanning in Belief Space [1] proposed a dynamic replanning scheme in belief space that enables online replanning for simultaneous localization and planning (SLAP). The underlying partially observable Markov decision process (POMDP) introduced by SLAP is approximated offline based on the Feedback-based Information RoadMap (FIRM) method, which provides a reliable framework for solving the problem of motion planning under uncertainty by reducing the intractable dynamic programming (DP) to a tractable DP over the nodes of the FIRM graph. Furthermore, this paper extends previous work on FIRM by proposing a belief space planning scheme by extending rollout policy methods to the stochastic partially observable setting. In particular, this method forces the system to globally replan at every time step to enable SLAP. The extensive simulation and experimental results shows that the proposed planner recovers the performance trade off in the stabilization phase of FIRM.

In this paper, a nonholonomic robot—the iRobot Create—is used in the experiments and it is modeled as a unicycle. To steer toward the target node, controllers designed for stabilizing nonholonomic systems are to be used. A polar coordinate-based controller and a dynamic feedback linearization-based controller are implemented based on *Steering with Sinusoids*.

References

[1] A. Agha-mohammadi, S. Agarwal, S. Kim, S. Chakravorty, and N. M. Amato. Slap: Simultaneous localization and planning under uncertainty via dynamic replanning in belief space. *IEEE Transactions on Robotics*, 34(5):1195–1214, Oct 2018.

[2] Muhammad Rehan and M Reyhanoglu. Motion planning for a knife-edge moving on the surface of a torus. October 2018.