DP2: Differential Drive Robot in Artificial Gravity

Varsha Krishnakumar and Ajitesh Muppuru
University of Illinois at Urbana-Champaign

The objective of this exploration is to design, implement, and test a controller that enables a differential-drive robot to move quickly around the inside of a rotating space station under artificial gravity. The controller and simulation are implemented, designed, and tested in a Pybullet Jupyter Notebook environment.

I. Nomenclature

ODE	=	Ordinary Differential Equation
v	=	Forward speed (m/s)
$\dot{v}$	=	Forward acceleration (m/s^2)
$\dot{\omega}$	=	Heading angle change rate (rad/s^s)
$\ddot{\theta}$	=	Pitch acceleration (rad/s^2)
$\dot{e}_{lateral}$	=	Lateral error
$\dot{e}_{heading}$	=	Heading error
$\dot{v}$	=	Forward acceleration (m/s^2)
f	=	Equations of motion as a function
θ	=	Pitch angle (rad)
$\dot{\theta}$	=	Pitch rate (rad/s)
$\ddot{\theta}$	=	Pitch acceleration (rad/s^2)
$\dot{e_{lateral}}$	=	Lateral error
$\dot{e_{heading}}$	=	Heading error
ω	=	Turning rate (rad/s)
$\dot{\omega}$	=	Heading angle change rate (rad/s^s)
τ_L	=	Left wheel torque (N $\cdot m$)
τ_R	=	Right wheel torque (N $\cdot m$)
u	=	Input vector
LQR	=	Linear Quadratic Regulator

II. Introduction

The differential-drive robot moves around the inside of a rotating space station under artificial gravity, given by the equation:

$$g = \omega_{station}^2 / r_{station} \tag{1}$$

Here, $\omega_{station}$ is the angular velocity of the rotating space station, and $r_{station}$ is the radius of the station (or, the distance from the center of rotation to its inner surface).
The robot consists of a chassis, a left wheel, and a right wheel. The robot can apply torques to its wheels to control itself, and the controller to be implemented will control it using a linearized version of the system's equations.

III. Theory

A. Equations of Motion for the Differential-Drive Robot System

In this system, conventionally, the robot is under the influence of artificial gravity induced by the rotation of the space station. However, due to the complex nature of the problem, it will be assumed that the inside of the space station is flat. In this case, the system is termed as a *nonholonomic* system. While the constrained nonlinear equations of

motion are derived using Lagrangian mechanics, the unconstrained equations of motion are obtained using the nullspace of the constraint matrices[†]

The Robot system is modeled based on a model in state-space form. [‡] The motion of this particular system is governed by the form of Ordinary Differential Equations below.

$$\begin{bmatrix} \frac{w_y \sin(\phi)+w_z \cos(\phi)}{\cos(\theta)} \\ w_y \cos(\phi) - w_z \sin(\phi) \\ w_x + w_y \sin(\phi)\tan(\theta) + w_z \cos(\phi)\tan(\theta) \\ -\frac{55\sqrt{2}\tau_1}{1484} + \frac{55\sqrt{2}\tau_2}{1484} - \frac{150 w_y w_z}{371} \\ -\frac{55\sqrt{2}\tau_3}{1484} + \frac{55\sqrt{2}\tau_4}{1484} + \frac{150 w_x w_z}{371} \\ -\frac{55\sqrt{2}(\tau_1+\tau_2+\tau_3+\tau_4)}{2084} \end{bmatrix} \dot{x} = \begin{bmatrix} \dot{e}_{lateral} \\ \dot{e}_{heading} \\ \dot{v} \\ \dot{w} \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = f(e_{lateral}, e_{heading}, v, w, \theta, \dot{\theta}, \tau_R, \tau_L) \tag{2}$$

In order to design a proper controller, the system must be modeled in state-space form. The equations of motion must first be linearized in order to approximate the system with the desired state-space model. Correspondingly, the details of the function f are noted below.

$$\dot{x} = f = \begin{bmatrix} v\sin(e_{heading}) \\ \omega \\ -\frac{2400\tau_L+2400\tau_R+2808(\dot{\theta}^2+\omega^2)\sin(\theta)+65(50\tau_L+50\tau_R-39\omega^2\sin(2\theta)-900\sin(\theta))\cos(\theta)}{11700\cos^2(\theta)-12168} \\ \frac{32(-875\tau_L+875\tau_R-1443\dot{\theta}\omega\sin(2\theta)-2925 v\omega\sin(\theta))}{13(3120\sin^2(\theta)+2051)} \\ \frac{5(8450\tau_L+8450\tau_R-6591\omega^2\sin(2\theta)+60(100\tau_L+100\tau_R+117(\dot{\theta}^2+\omega^2)\sin(\theta))\cos(\theta)-152100\sin(\theta))}{1404(25\cos^2(\theta)-26)}[10pt] \end{bmatrix} \tag{3}$$

The input vector, u, can be modelled using τ_L and τ_R as entries.

$$u = \begin{bmatrix} \tau_L \\ \tau_R \end{bmatrix} \tag{4}$$

The equilibrium points for which to linearize the system about is the set of variables for f that cause $\dot{x} = 0$, and thus x remains constant. These equilibrium values are denoted with the same nomenclature as their state variables with the addition of the subscript e.

The state and input are defined based on our choice of equilibrium values:

$$\mathbf{x} = \begin{bmatrix} e_{lateral} - e_{lateral_e} \\ e_{heading} - e_{heading_e} \\ v - v_e \\ \theta - \theta_e \\ \dot{\theta} - \dot{\theta}_e \\ \omega - \omega_e \\ e_{lateral} - e_{lateral_e} \\ e_{heading} - e_{heading_e} \end{bmatrix} \tag{5}$$

$$\mathbf{u} = \begin{bmatrix} \tau_L - \tau_{L_e} \\ \tau_R - \tau_{R_e} \end{bmatrix} \tag{6}$$

The matrices A and B may then be found by calculating the Jacobian of f with respect to the state vector x and the input vector u. These were evaluated at the chosen equilibrium points v_e, θ_e, $\dot{\theta}_e$, ω_e, $e_{lateral_e}$ and $e_{heading_e}$.

[†]Tuttle, J. T. L., "Studies of systems with nonholonomic constraints: The Segway and the Chaplygin Sleigh," Purdue e-Pubs Available: https://docs.lib.purdue.edu/open_access_theses/386/.

[‡]https://tbretl.github.io/ae353-sp22/projectsthe-system

$$\mathbf{A} = \left.\frac{\partial f}{\partial x}\right|_{(v_e, \theta_e, \dot{\theta}_e, \omega_e, e_{lateral_e}, e_{heading_e})} \tag{7}$$

$$\mathbf{B} = \left.\frac{\partial f}{\partial u}\right|_{(v_e, \theta_e, \dot{\theta}_e, \omega_e, e_{lateral_e}, e_{heading_e})} \tag{8}$$

B. Controllability

Prior to implementing a working controller, the linearized system's controllability needed to be verified. This was accomplished by generating the controllability matrix, W. The procedure of outputting this matrix is detailed in the *DeriveEOM.ipynb* in the main repository. [§] This matrix is reliant on matrices A and B, and was verified to be full rank. Therefore, the system was deemed to be controllable.

C. Linear Quadratic Regulator (LQR)

Initially, K values were generated using the *Place Poles* method from SciPy's Signal Package. While this tool generated gain matrices, it did not allow for comprehensive tweaking of the controller. In this system, there is a need for precise preservation of some of the states, given that the primary requirement is keeping the robot on the desired path. Therefore, the *LQR* (Linear Quadratic Regulator) method was soon used in place of the Place Poles tool. This primarily aids in close *tuning*: allowing for some states to be more closely controlled than others.

LQR works by minimizing the input ($u_{[t_0,\infty}$) subject to the following equations:

$$\int_{t_0}^{\infty} \left(x\left(t\right)^T Q x\left(t\right) \; + \; u\left(t\right)^T R u\left(t\right) \right) dt \tag{9}$$

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{10}$$

$$x(t_0) = x_0 \tag{11}$$

The Scipy *Linalg* library's *solve_continuous_are()* takes care of minimizing the above equations and also calculates the minimizing gain matrix, K. The LQR controller takes in the controller matrices A and B, but also takes in inputs for control weights called Q and R. Q and r tuned by adjusting the diagonal entries as shown below.

$$Q = \begin{bmatrix} q_1 & 0. & 0. & 0. & 0. & 0. \\ 0. & q_2 & 0. & 0. & 0. & 0. \\ 0. & 0. & q_3 & 0. & 0. & 0. \\ 0. & 0. & 0. & q_4 & 0. & 0. \\ 0. & 0. & 0. & 0. & q_5 & 0. \\ 0. & 0. & 0. & 0. & 0. & q_6 \end{bmatrix} \tag{12a}$$

$$R = \begin{bmatrix} r_1 & 0. \\ 0. & r_2 \end{bmatrix} \tag{12b}$$

The most efficient method of tuning is adjusting these entries relative to each other so that different weights can be placed to control each state. Moreover, a larger relative value for a particular state results in a larger deviation from equilibrium. In the system of the Segway Robot, $e_{lateral}$ is of primary importance, as noted in the primary requirements; deviating too far from the center line of the desired path can result in an undesired system.

[§] https://github.com/varshakrishnakumar/AE-353-Design-Project-2

IV. Experimental Methods

A. Requirements/Verification

For this project, success will be determined quantitatively by various parameters that we have before completing the experiments.

Requirements:

1) The robot shall maintain a heading deviation between ± 0.1 radians from the center line of the path.
2) The robot shall maintain a lateral deviation between ± 0.1 meters from the center line of the path.
3) The robot shall not exceed a pitching angle of ± 0.2 radians.
4) The robot shall maintain criteria 1, 2, 3 on a bumpy terrain.
5) The robot shall maintain criteria 1, 2, 3 with the station traveling at a velocity of -0.5 m/s.
6) The robot shall maintain criteria 1, 2, 3 without falling/stopping for at least 30 seconds.

In addition to the performance requirements set, we also set the equilibrium points the robot should be converging toward.

1) With stability in mind, $e_{lateral_e}$, $e_{heading_e}$, ω_e, θ_e, $\dot{\theta_e}$, τ_{L_e}, and τ_{R_e} shall be set to 0. This is based on the ideal state of the robot moving completely upright, perfectly centered at the given radius, using minimal torque to stabilize itself over the duration of the run.
2) Since the other equilibrium points are all set to 0, the value for the equilibrium point of the forward speed can hold any value. The group chose v_e = 1.2 m/s.

Verification: A Jupyter Notebook using a Pybullet environment was utilized as code interface for simulating and testing the Differential-Drive Robot. The instructions for compilation of the project code is detailed in the GitHub repository associated with this report. * Instead of verifying random gain matrices, gain matrices will be found by simulating various Q and R matrices and their subsequent gain matrices and verifying that they fulfill the requirements. These Q and R matrices will be found by iterating through higher Q components based on the behavior desired (i.e. q1 : lateral error). After obtaining a list of possible gain matrices, the right gain matrices can be found by testing to see if they achieve an active runtime of 60 seconds without falling or stopping. The implementation of this can be viewed in the code associated with the report.

V. Results

A. LQR Matrix Selection

The next iteration more comprehensively selected Q components to focus on stability which meant reducing lateral and heading error. In light of these concerns, Q(0,0) and Q(1,1) were increased, leaving other values fairly low. The following matrices reflect the implementation of these considerations.

$$Q = \begin{bmatrix} 7631.5790 & 0. & 0. & 0. & 0. & 0. \\ 0. & 368.447 & 0. & 0. & 0. & 0. \\ 0. & 0. & 0.0763 & 0. & 0. & 0. \\ 0. & 0. & 0. & 0.0763 & 0. & 0. \\ 0. & 0. & 0. & 0. & 0.0763 & 0. \\ 0. & 0. & 0. & 0. & 0. & 3.7105 \end{bmatrix} \tag{13a}$$

$$R = \begin{bmatrix} 1.5 & 0. \\ 0. & 1.5 \end{bmatrix} \tag{13b}$$

These inputs yielded the following gain matrix.

$$K = \begin{bmatrix} 50.437 & 27.247 & -0.5044 & 5.1187 & -10.807 & -1.3270 \\ -50.437 & -27.247 & -0.5044 & -5.1187 & -10.807 & -1.3270 \end{bmatrix} \tag{14}$$

*https://github.com/varshakrishnakumar/AE-353-Design-Project-2

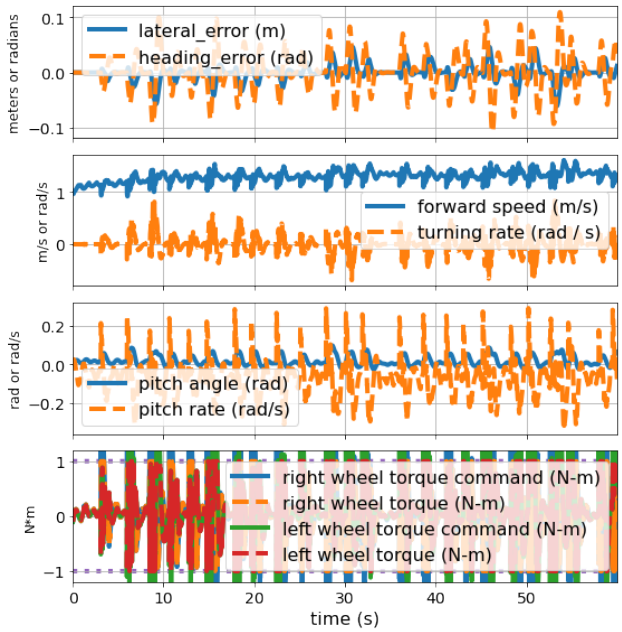

Fig. 1 Segbot Performance Using Controlled Q for LQR weights

The gain matrix that resulted from this new iteration yielded a fairly stable Segbot that fulfilled the desired requirements. Furthermore, the Segbot was able to last at least a lap on the modified version of the track with no bumps. On a bumpy track, the Segbot was able to almost complete a lap, but terminated its movement a little before the end of the track.

B. Testing Range of Initial Conditions

In order to test the efficiency of the controller, the model was tested with varying initial conditions including initial forward speed (v_i) and initial pitch angle (θ_i). This method of testing was implemented to reveal the true theoretical limits of the efficiency of the controller.

The most critical outcomes from this simulation and corresponding testing are:

1) For the initial condition of v_i = 1.2 m/s and an equilibrium condition of v_e = 1 m/s, the Segbot performs well in terms of reaching the desired requirements. Notably, as seen in Figure 2, the lateral and heading error stabilize and remain less than 0.1 m and rad respectively. Additionally, the pitch angle remains less than 0.2 rad.
2) From Figure 3, it can be seen that the pitch angle has to be 0 rad in order for the Segbot to meet the set requirements. This is likely due to the fact that a slight tilt in the orientation of the robot can
3) The final resulting limits to the initial conditions were observed as: v_i = [0, 1.2] and θ_i = [0]

As seen from the results, the controller is efficient in stabilizing for initial conditions that are close to the equilibrium or desired conditions. It must be noted that inside of the space station was assumed to be flat when in reality, it is supposed to be curved upward. This assumption influenced the model of the controller, which could have resulted in the limitations to the set initial conditions. Therefore, initial conditions that are further away from the selected equilibrium conditions are not practically feasible.

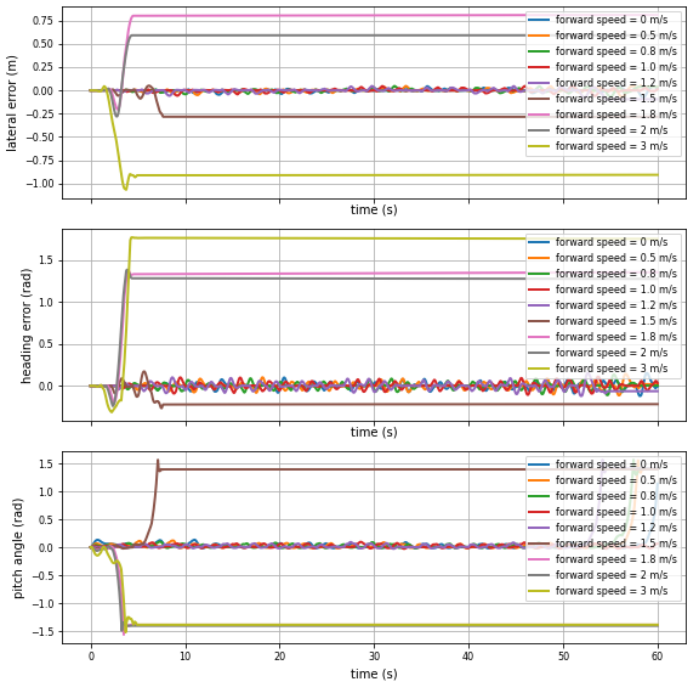

Fig. 2 Performance Based on Varying Initial Forward Speeds

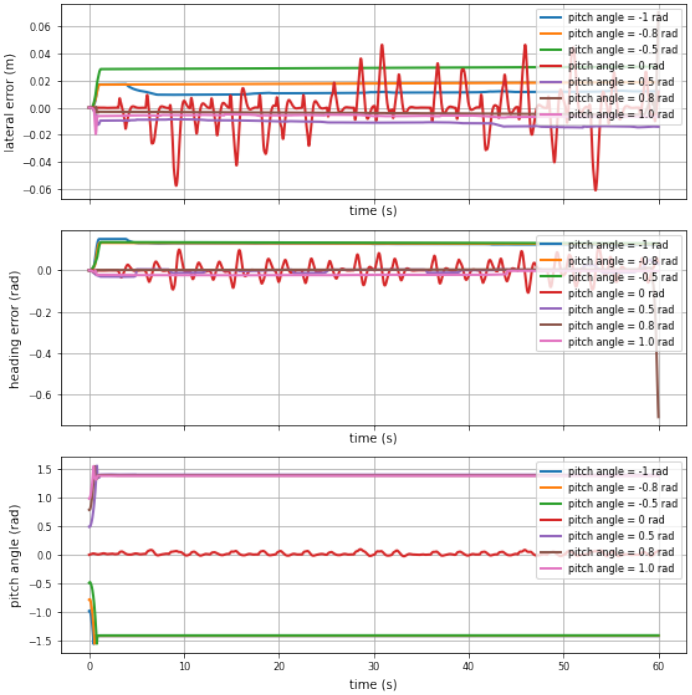

Fig. 3 Performance Based on Varying Initial Pitch Angles

VI. Conclusion

The designed Differential-Drive Robot model is efficient in achieving the desired equilibrium conditions and staying on the circular track by attaining stability appropriately. The utilization of the LQR method was particularly useful in the generation of the K gains matrix which was crucial in testing the efficiency of the system. Although the system loses the ability to attain stability as the initial conditions stray further away from the equilibrium conditions, it must also be noted that the space station was assumed to be flat, and the theoretical model was only an estimation of the problem.

Acknowledgments

Majority of the driver code for the implementation of the Robot model was provided in lecture in AE 353 at the University of Illinois conducted by Professor Timothy Bretl and the Teaching Assistant, Mr. Jacob Kraft. The guided learning on the theory of control systems and coding techniques assisted in the modelling and implementation of this project. Furthermore, the utilization of *Campuswire* as a tool for query clarification further improved theoretical understanding.

References

[1]: Tuttle, J. T. L., "Studies of systems with nonholonomic constraints: The Segway and the Chaplygin Sleigh," Purdue e-Pubs Available:https://docs.lib.purdue.edu/open$_access_theses$/386/

[2]: "Projects," AE 353: Aerospace Control Systems Available: https://tbretl.github.io/ae353-sp22/projectsthe-system.

[3]: Muppuru, A., and Krishnakumar, V., "Varshakrishnakumar/ae-353-design-project-2: Designing and implementing a Differential-Drive Robot," GitHub Available https://github.com/varshakrishnakumar/AE-353-Design-Project-2.

Appendix

Day	Task	Person or People
02/23/2022	Initiation and set-up of report document	Varsha Krishnakumar
02/25/2022	Basic structure for Abstract and Introduction sections	Ajitesh Muppuru
02/25/2022	Set up of A, B and D sections of Theory	Varsha Krishnakumar
02/25/2022	Set up of C section of Theory	Ajitesh Muppuru
02/25/2022	Set up of GitHub Repository with relevant code and testing various initial and equilibrium conditions for running simulations	Varsha Krishnakumar
02/25/2022	Set up of EOM and segbot Demo code for running simulations	Ajitesh Muppuru
03/03/2022	Writing algorithm to test various LQR	Varsha Krishnakumar
03/04/2022	Debugging LQR Loop to achieve functionality within Segbot file	Ajitesh Muppuru
03/05/2022	Writing Experimental Methods Section	Varsha Krishnakumar
03/05/2022	Finding usable data to represent our controller	Varsha Krishnakumar
03/05/2022	Writing Results Section	Ajitesh Muppuru
03/09/2022	Finalizing K-Value Selection through LQR	Varsha Krishnakumar and Ajitesh Muppuru
03/09/2022	Creating plots for 16 different initial conditions	Varsha Krishnakumar
03/10/2022	Implementing Feedback to Report and Condensing Experimental Methods Section	Ajitesh Muppuru
03/11/2022	Making Final Code Notebook Presentable	Varsha Krishnakumar
03/11/2022	Writing Script for Video	Ajitesh Muppuru and Varsha Krishnakumar
03/11/2022	Finishing Results Section and Writing Conclusion	Varsha Krishnakumar
03/11/2022	Recording/Assembling Video	Ajitesh Muppuru