DP2: Control of a Differential-Drive Robot in Artificial Gravity

Aden Litwiller and Patrick Thornton
University of Illinois at Urbana-Champaign

The objective of this...

I. Nomenclature

ODE	=	Ordinary Differential Equation
x	=	state matrix
u	=	input matrix
e_{lateral}	=	lateral error (m) to the left, or the distance from the wheel center of the robot to the centerline of the station ring
e_{heading}	=	heading error (rad) to the left, which is the difference between the orientation of the robot and the direction of the station r
v	=	forward speed (m/s) of the robot
ω	=	turning rate (rad/s) to the left
θ	=	forward pitch angle (rad)
ϕ	=	forward pitch rate (rad/s)
τ_R	=	forward right wheel torque (Nm) applied to the right wheel by the chassis
τ_L	=	forward left wheel torque (Nm) applied to the left wheel by the chassis

II. Introduction

Introduction...

III. Theory

A. Equations of Motion for the Differential-Drive Robot in Artificial Gravity

In the process of designing a proper controller, the description of the dynamic system had to be linearized before it could be placed into state-space form. Given a set of ODEs, the system first had to be rewritten in a way such that any second-order ODEs were replaced by a set of two first order ODEs. To do so, the arbitrary variable ϕ was employed as follows:

$$\phi = \dot{\theta} \tag{1a}$$

$$\dot{\phi} = \ddot{\theta} \tag{1b}$$

The modified system could then be written as

$$\begin{bmatrix} \dot{e}_{\text{lateral}} \\ \dot{e}_{\text{heading}} \\ \dot{v} \\ \dot{\omega} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = f(e_{\text{lateral}}, e_{\text{heading}}, v, \omega, \theta, \phi, \tau_R, \tau_L) \tag{2}$$

With the following function f:

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

Next, a valid equilibrium point (i.e., the set of variables in f that make $f = 0$) had to be determined analytically:

1) $\omega = 0$ as is apparent from the second equation in f.
2) $\phi_e = \dot{\theta}_e = 0$ as is apparent from the fifth equation in f.
3) $\theta_e = 0$, which eliminates all of the $\sin(\theta)$ terms while ensuring that the robot maintains an upright position.
4) $e_{\text{heading},\,e} = 0$, as required by the first equation and the choice of a non-zero v_e.
5) $e_{\text{lateral},\,e} = 0$. This is not explicitly required by the equations in f, but any non-zero value was specified as being too far from the centerline of the station ring.
6) $\tau_{R_e} = \tau_{L_e} = 0$, as is required after numerically evaluating the rest of the equations with the equilibrium variables above.
7) These variable definitions are enough to make $f = 0$. This indicates that v_e, the desired velocity for the robot to maintain, may have any have any real value. With the goal of maximizing the speed at which the robot traverses the ring of the space station, a target velocity of $v_e = 1$ was set.

The state and input were then defined based on this choice of equilibrium point:

$$x = \begin{bmatrix} e_{\text{lateral}} - e_{\text{lateral},\,e} \\ e_{\text{heading}} - e_{\text{heading},\,e} \\ v - v_e \\ \omega - w_e \\ \theta - \theta_e \\ \phi - \phi_e \end{bmatrix} = \begin{bmatrix} e_{\text{lateral}} - 0. \\ e_{\text{heading}} - 0. \\ v - 4. \\ \omega - 0. \\ \theta - 0. \\ \phi - 0. \end{bmatrix} \tag{4a}$$

$$u = \begin{bmatrix} \tau_R - \tau_{R_e} \\ \tau_L - \tau_{L_e} \end{bmatrix} = \begin{bmatrix} \tau_R - 0 \\ \tau_L - 0 \end{bmatrix} \tag{4b}$$

The matrices A and B were then found by calculating the Jacobian of f with respect to the state vector x and the input vector u, and were evaluated at the chosen equilibrium points defined above.

$$A = \left.\frac{\partial f}{\partial x}\right|_{(x_e,u_e)} = \begin{bmatrix} 0. & 4. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 1. & 0. & 0. \\ 0. & 0. & 0. & 0. & -125. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 1. \\ 0. & 0. & 0. & 0. & 541.66666667 & 0. \end{bmatrix} \tag{5a}$$

$$B = \left.\frac{\partial f}{\partial u}\right|_{(x_e,u_e)} = \begin{bmatrix} 0. & 0. \\ 0. & 0. \\ 12.07264957 & 12.07264957 \\ 1.05014439 & -1.05014439 \\ 0. & 0. \\ -51.46011396 & -51.46011396 \end{bmatrix} \tag{5b}$$

After finding the A and B matrices, the system was finally put into state-space form:

$$\dot{x} = Ax + Bu \tag{6}$$

B. Controller Implementation

The first step in implementing a controller is to test that the system is controllable in theory. In order for a system to controllable, its controllability matrix W must be full rank, or in other words, the rank of W must equal the number of states. Python implementation was used to determine that this system's controllability matrix is full rank, thus confirming that it is controllable.

The input matrix u was then rewritten as below in order to implement a closed-loop feedback system. Here, K is some constant gains matrix, and the form implies that the input depends on the state.

$$u = -Kx \tag{7}$$

With this definition of u, the system may be rewritten as:

$$\dot{x} = (A - BK)x = Fx \tag{8}$$

In order to achieve a reliable controller, the resulting matrix F must have have eigenvalues, s, with real parts that are all negative. However, it is apparent that not all valid K matrices yield the desired robot behavior. For example, the actuator can only apply torques up to a maximum of ±1 Nm, but some K matrices that work in theory will provide greater torques than is allowed. To determine a working K matrix for simulation purposes, the Linear Quadratic Regulator (LQR) optimal control problem below was applied by computing K and gathering simulation data for various combinations of $Q \in [0.01, 0.3)$ and $R \in [4., 12.)$. Simulations were run for 20 seconds each, and the plots shown in *Fig. 1* were produced to show the effects of Q and R on final simulation measurements.

$$\underset{u_{[t_0,\infty)}}{\text{minimize}} \qquad \int_{t_0}^{\infty} \Big(x(t)^T Q x(t) + u(t)^T R u(t) \Big)\, dt \tag{9a}$$

$$\text{subject to} \qquad \dot{x}(t) = Ax(t) + Bu(t), \quad x(t_0) = x_0 \tag{9b}$$

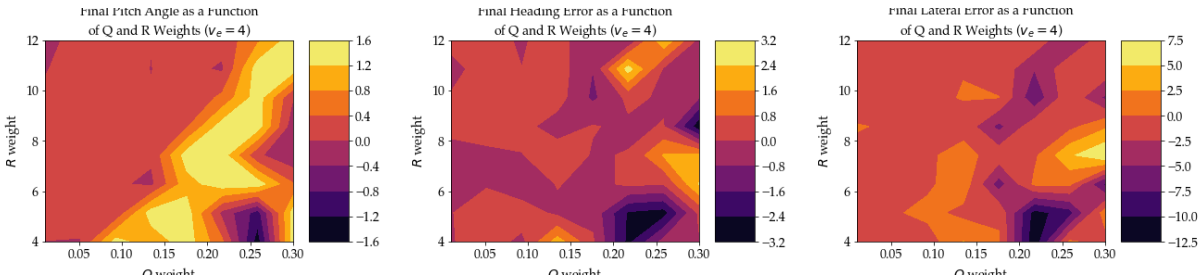

Fig. 1 Contour plots of (a) θ_{final} (b) $e_{\text{heading, final}}$ (c) $e_{\text{lateral, final}}$

$0.01 < \theta_{\text{final}} < 0.01$ was used as the criteria to determine effective K matrices that kept the robot in a vertical orientation at the conclusion of 20 seconds, and $Q = 0.01$ and $R = 4.$ were chosen to generate the K matrix with the corresponding eigenvalues matrix shown below using the LQR method.

$$K = \begin{bmatrix} 0.03535534 & 0.42630818 & -0.03535534 & 0.63812378 & -10.53251188 & -0.46215797 \\ -0.03535534 & -0.42630818 & -0.03535534 & -0.63812378 & -10.53251188 & -0.46215797 \end{bmatrix} \tag{10a}$$

$$s = \begin{bmatrix} -0.66548 & -0.33737 + 0.57663\,j & -0.33737 - 0.57663\,j & -25.21446 & -21.48332 & -0.01394 \end{bmatrix} \tag{10b}$$

IV. Experimental Methods

A. Requirements

The controller shall be designed such that the robot shall operate for 20 seconds at a desired velocity of 1 m/s $\leq v \leq$ m/s while θ remains within $\pm\frac{\pi}{12}$ rad, e_{lateral} remains within ±0.75 m, and e_{heading} remains within $\pm\frac{\pi}{4}$ rad. These requirements must be met for initial conditions v_i, θ_i, $e_{\text{lateral, i}}$, and $e_{\text{heading, i}}$ that fall within these ranges. These requirements must also be satisfied for the case when the station ring is spinning at its nominal rate of −0.5 m/s.

B. Verification

The requirements for the controller were verified using a PyBullet simulation. Python code was written to compute the equations, choose an equilibrium point, derive a state-space model, and define a controller that implemented linear state feedback. The data describing the system was generated with the simulation and exported into a Jupyter notebook to be analyzed with Python. Then, e_{lateral}, θ, and v were recorded and plotted at each time step. The maximum e_{lateral}, θ, and v for the last 20 seconds of the simulation will be recorded. If these maximum values are within ±0.1 m, ±0.1 rad, and 4 ±0.1 m/s respectively, then all requirements will have been met.

C. Trials

First, for initial conditions set at the equilibrium point, measurements from the simulation were plotted for each variable over time. To establish limits of performance, the verification was executed multiple times, varying θ_0 from -0.4 rad to 0.4 rad with other initial variables set to their equilibrium points. Next, the final θ at the end of the 20 second simulation was plotted over the range of the initial conditions. For one of these trials, three plots were generated that show, over the twenty seconds of the simulation, e_{lateral} and e_{heading}, forward speed and turning speed, pitch angle and pitch rate, and right and left wheel torque commanded and provided. Lastly, a method was implemented to report if any requirements were not met within the required limits of performance and to list any initial condition that caused the robot to fail any requirement.

V. Results and Discussion

A. Controller Verification in Simulation

The results of the controller verification simulation for the default initial conditions are shown below in *Figure 2*. The most critical observations from this simulation are as follow:

1) $v_e = 4$ m/s, but the robot fails to exceed $v > 2$. However, the data does show that v slowly increases over the 20 second simulation period. This failure may be due to an improper choice of K, or it may also be due to the system's torque limitations and priority of maintaining stability. Further testing would reveal the cause. Additionally, oscillations can be observed which would indicate each time the robot encounters a bump on the surface of the station ring.
2) The controller does an excellent job of maintaining an upright position ($\theta \approx \theta_e = 0$) during the 20 second period. Slight oscillations in θ are present, but those are likely explained by the aforementioned bumps.
3) e_{lateral} and e_{heading} initially maintain their equilibrium values of 0, but it appears that they significantly deviate from equilibrium after encountering a bump. Over the 20 second period, however, it does appear that the controller attempts to correct these errors within the physical limitations of maintaining stability.
4) The wheel torque commands oscillate wildly as a result of encountering bumps and attempting to return the robot to equilibrium values. While the torque commands approach the limits of $\tau_{limit} = \pm 1$ Nm, they do not exceed these values. If the simulation was run for a longer amount of time, this may not be the case as the magnitude of these oscillations appear to become more significant as time, and thus v at impact of a bump, increase.

B. Establishing Limits of Performance and Verification of Requirements

The measurements of θ_{final} from running simulations for varying θ_i over a period of 20 seconds is shown below in *Figure 3*.

As observed in *Figure 3*, the controller is effective in stabilizing the robot for θ_i that are close to $\theta_e = 0$. Specifically, the limit on θ_i is approximately $\theta_{\text{limit}} = \pm\frac{\pi}{12}$. For θ_i outside of this range, the system cannot maintain stability due to the

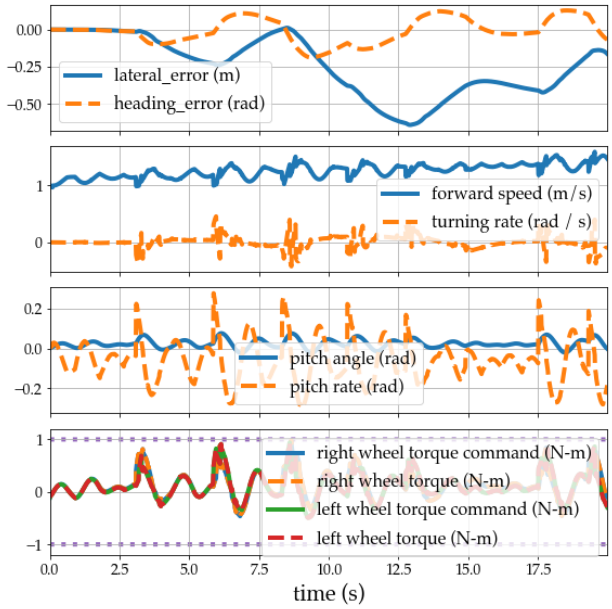

Fig. 2 Results of Simulation

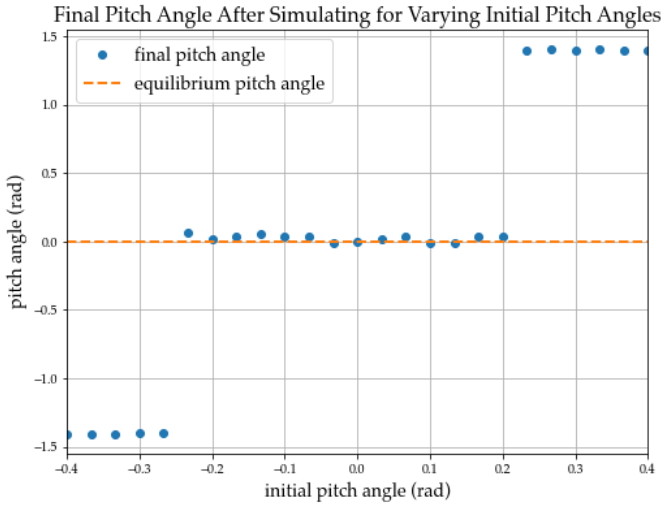

Fig. 3 θ_{final} From Simulations of Varying θ_i

limitations on τ.

For the range of varying θ_i, the verification results are presented below in *Table 1*.

θ_i	**Equilibrium Measurement Exceeded**	**Measurement Value**
0.06666	Lateral Error	-0.75035
0.19999	Lateral Error	-0.75029
0.23333	Pitch Angle	0.26858
0.29999	Heading Error	0.78699

Table 1 θ_i for which Verification Failed

From *Table 1*, the latter three entries appear to show a trend that values of θ_i close to the limits of performance have a higher likelihood of failing to meet the defined requirements. This is likely due to the controller having a tendency to over-correct in its attempt to maintain stability, as well as a result of the bumps in the station ring surface. However, it is interesting that $\theta_i = 0.06666$ failed verification, as it is relatively close to $theta_e = 0$. This result demonstrates that there is random variability in the simulation that should be further examined through subsequent trials, or possibly tested with a refined choice of K.

VI. Conclusion

Conclude

VII. Acknowledgments

Acknowledgements

VIII. Appendix

Day	Task	Person or People
02/22/2022	Creation and basic organization of Overleaf report document	Patrick Thornton
02/23/2022	Met to discuss direction of the project and roughly delegate tasks for the first draft, determined valid equilibrium point	Aden Litwiller & Patrick Thornton
02/23/2022	Rewrote the provided equations of motion as system of first-order ODEs	Aden Litwiller
02/23/2022	Began coding a function to return a state-space model for a given choice of equilibrium point	Patrick Thornton
02/25/2022	Initialized GitHub repository, completed state-space model function, implemented code to determine K from chosen poles, completed section (A) of Theory	Patrick Thornton
03/03/2022	Tested that the system was controllable in theory through Python and described the process in "Theory"	Aden Litwiller
03/03/2022	Discussed how to choose K using LQR method	Aden Litwiller & Patrick Thornton
03/03/2022	Wrote requirements and verification methodology	Aden Litwiller
03/03/2022	Began writing code to refine choice of K using the LQR method by plotting simulation data for various choices of Q and R	Patrick Thornton
03/04/2022	Completed code to determine a valid K using LQR and revised "Theory" section to reflect changes	Patrick Thornton
03/04/2022	Wrote description of the trials that were executed	Aden Litwiller
03/04/2022	Wrote "Results and Discussion" section: incorporated plots and a table generated with Python, then provided discussion of said results	Patrick Thornton
03/04/2022	Adjusted "Requirements" to more closely match the current code	Patrick Thornton