A. James Clark
SCHOOL OF ENGINEERING

University of Maryland, College Park

Control of Robotic Systems

Simulation and Analysis of the Ballbot

Members: *UID:*
Shubham Sonawane 116808996
Prasheel Renkuntla 116925570

Contents

Acknowledgement

We would like to show our deepest gratitude to Dr. Umashankar Nagarajan, Staff Research Scientist, Toyota Research Institute, for giving us an immediate and compelling response for our project on his paper. We would also like to thank our Professor, Dr. Waseem Malik, Control of Robot Systems, University of Maryland, College Park, whose constant and apt guidance on the project helped us overcome the challenges faced.

Lastly, We would also like to thank all our friends, family, fellow students, and faculty who were involved directly or indirectly towards the completion of this project.

Abstract

The Ballbot is an omnidirectional balancing mobile robot. Due to its single point of contact design, it is dynamically stable. It has a high center of gravity due to its tall and narrow structure when compared to other statically stable mobile robots.

This project discusses the source's [4] approach to the design and development of the dynamic model and control architecture of the robot. The project analyses a trajectory planning algorithm for various motions. It verifies the results presented on the source with thorough analysis. This project also presents the simulation results obtained with our understanding of the source. Additionally, a circular trajectory is simulated to verify the results and its generalized applications. The robot was simulated on MATLAB, SIMULINK and V-REP.

List of Figures

List of Tables

1 Introduction

The ballbot is a single wheel omnidirectional balancing mobile robot. It is a human friendly robot due to its tall and narrow structure with a small footprint. Thus, it has a design favourable to navigate and interact through constricted human environments. It is based on four-wheel inverse mouse-ball drive to move around and has full rotation about its vertical axis due to its Yaw mechanism. The ballbot was first introduced in 2005 in [2] and gained recognition in 2006 with [3]. Though, it produced undesired jerky motions upon tracing a desired ball trajectory but the controller did balance the robot efficiently. Further, more designs were produced, this being one of them.

This report analyses the design and control of the ballbot, to study its characteristics and motion inside a human interactive environment. The drive system of the ballbot consists of four rolling motors to overcome the hopping motion from the earlier designs. The control mechanism of the ballbot is collectively made of a balancing control, a station keeping control, velocity control, yaw control and legs adjustment control. A combination of balancing and legs adjust control allows the ballbot to have a smooth transition from a dynamically stable state to statically stable state (introduced further ahead in the report). The balancing control has the capacity to resist large and sudden forces, collisions to walls, and etc.

This report also talks about the trajectory planning of the robot which when combined with the balancing control, allows a smooth motion to track a desired trajectory without any jerks. These are shown with the simulation results from MATLAB, SIMULINK and V-REP.

1.1 Background

Initially, the concept of balancing wheeled robot, i.e, cart-pendulum system was the canonical problem in controls. Popularity to two wheel robots was achieved after the introduction of Segway RMP [6]. Later, a variety of two wheeled robots arrived in the market but with a disadvantage that they have a kinematic constraint, i.e, they cannot slip sideways. They balance only in one vertical plane, but a single wheel robot can balance in both vertical planes and has an omnidirectional motion. The ballbot was first introduced at Carnegie Mellon University in 2005, post which there were numerous other ballbots from all over the world like the Rezero, etc. The primary difference between the ballbot in the report and other ballbots from around the world is that of the rive system. Most of the other robots developed are omni-wheel drive based, whereas this robot is driven by a four roller mechanism. There is a slip ring assembly and a separate motor for unlimited yaw rotation. Thus, the ballbot has a greater advantage over the other robots with a high dynamic stability.

1.2 Motivation

Most of the mobile robots are designed to be inherently stable. The stability is achieved by keeping the robot in contact with the ground plane at 3 or more points, using linkages or wheels.The limitations of these robots are as follows:-

1. Due to the multiple contacts design, these robots in general occupy a larger surface area.

2. In order to have stability during high speed manoeuvres, the center of mass of the robots needs to be as close as possible to the ground surface.

3. If the height of the robot is large, there is a danger of tipping over during acceleration or deceleration.

All these limitations lead to the development of a ball balancing robot. Although it has its advantages over multiple wheeled robots, it requires a robust control architecture in order to account for its highly dynamical nature. This report addresses the control architecture for balancing, station keeping and velocity control. Also, trajectory planning for a particular motion of the BallRobot has been addressed in this report.

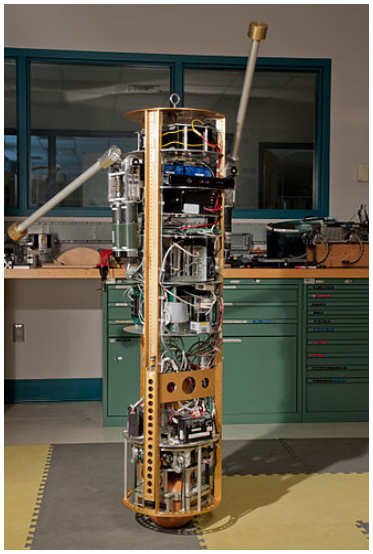

Figure 1: Ballbot, CMU

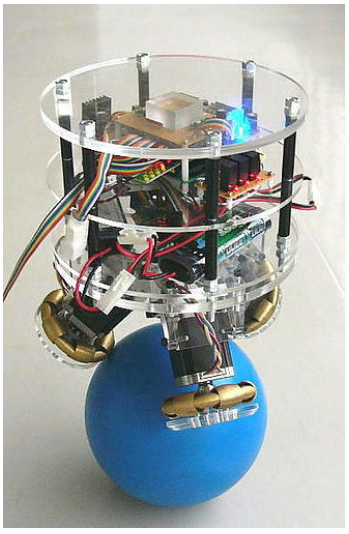

Figure 2: BalllP, University of Japan

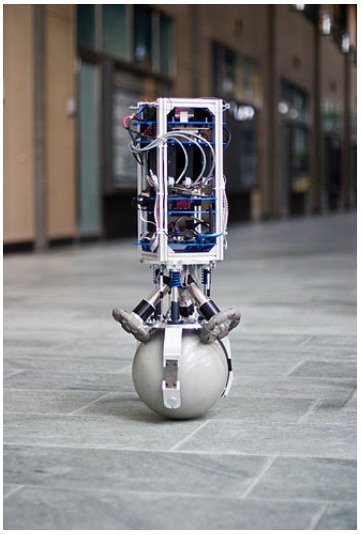

Figure 3: Rezero, ETH Zurich

Image source : https://en.wikipedia.org/wiki/Ballbot

2 System Description

The BallBot is a human sized robot, reaching the height of 1.41m. The ballbot can be segmented into 5 main sections, as follows:

1. Cylinder, or the upper body: The cylinder houses the electronic hardware necessary for controlling the motion of the robot. It also houses the sensors such as Inertial Measurement Unit (IMU) which is helpful in measuring the body angles (roll and pitch)

2. Inverse Mouse Ball Drive system: The ball bot is driven using an inverse mouse ball drive system. In a traditional mouse ball drive system, the ball rotates the rollers in the mouse which are used to move the pointer location on the screen. In the inverse mouse ball drive, the sphere is rotated by 4 rollers, each driven by a separate motor. The motor is coupled with an encoder, which measures the number of rotations of the rollers

3. Yaw Drive: A bearing is used to attach the yaw drive motor to the body. An encoder is used as feedback to keep track of the yaw angle. There are no bounds on the yaw angle attained by the robot.

4. Legs: In order to transfer the robot into the Statically Stable State, the robot is equipped with 3 legs. Each leg is operated by a separate DC motor, which has an encoder feedback. This ensures that the legs move the exact distance as desired. There are switches at the end of the legs, which detect contact with the floor. This feedback is necessary to stop the movement of the legs.

5. Sphere: The complete system rests on a sphere. The torque from the rollers helps move the sphere, thereby providing motion to the ballbot.

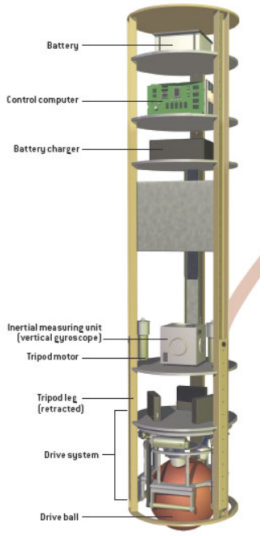

Figure 4: Components of Ballbot

Image source : http://www.electricalelibrary.com/en/2018/03/02/what-is-ballbot/

3 Dynamic Model of the Robot

The dynamic equation of the robot is modelled in 2D. Following assumptions were necessary while developing the dynamic equations:-

1.Motion in the XZ plane and the YZ plane is decoupled, and the equation of motion in these planes is same.

2.There is no slipping between the floor and the sphere.

3.There is no slipping between the rollers and the sphere.

4.All the bodies being modelled are rigid bodies i.e. there is no deformation.

3.1 Planar ballbot model

The ballbot has 5 degrees of freedom, two translational and 3 rotational. In the 2D model, we deal with the position and not the orientation. First, we define our world frame, which coincides with the start position of ball. The frame of reference being considered here is the XZ plane. As per our assumptions mentioned earlier, the equation of motion of the YZ plane will be the same as the XZ plane. Th body angle ϕ is given by the angle between the vertical axis and the body axis. The body axis is the line connecting the centre of mass of the ball to the center of mass of the body. θ denotes the angular configuration of the ball. The horizontal position of the ball can be established using these 2 state variables, and is given by $x_w = r_w(\theta + \phi)$.

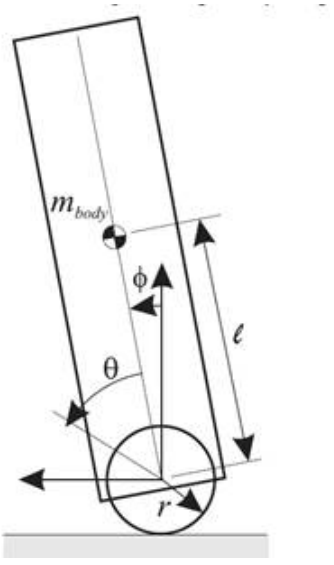

Figure 5: Planar Diagram

Image source : http://www.electricalelibrary.com/en/2018/03/02/what-is-ballbot/

The dynamic equation of the ballbot is derived by using the Euler-Lagrange equation, and is given as follows:

$$M(q)\ddot{q} + C(q,\dot{q}) + G(q) + D(\dot{q}) = \begin{bmatrix}\tau \\ 0\end{bmatrix} \tag{1}$$

$$q = \begin{bmatrix}\theta \\ \phi\end{bmatrix} \tag{2}$$

$$M(q) = \begin{bmatrix}\alpha & \beta \\ \alpha + \beta\cos\phi & \alpha + \gamma + 2\beta\cos\phi\end{bmatrix} \tag{3}$$

$$C(q,\dot{q}) = \begin{bmatrix}-\beta\sin\phi\dot{\phi}^2 \\ -\beta\sin\phi\dot{\phi}^2\end{bmatrix} \tag{4}$$

$$G(q) = \begin{bmatrix}0 \\ -\frac{\beta g\sin\phi}{r}\end{bmatrix} \tag{5}$$

$$D(\dot{q}) = \begin{bmatrix}D_c sgn(\dot{\theta}) + D_v\dot{\theta} \\ 0\end{bmatrix} \tag{6}$$

$$\alpha = I_w + (m_w + m_b)r_w^2;\ \beta = m_b * r_w * l_b;\ \gamma = I_b + m_b l_b^2; \tag{7}$$

The dynamic equation has been derived in the Appendix A-3. The table below shows the ballbot robot specifications that can be used to compute the values of coefficients in equation (7) above.

Table 1: Ballbot Robot Specifications

Parameter	**Symbol**	**Value**
Ball Radius	r_w	0.1058 m
Roller Radius	r_r	0.006335 m
Ball mass	m_w	2.44 kg
Ball Inertia	I_w	0.0174 $kg * m^2$
Body center of mass height	l_b	0.69 m
Roll Moment of Inertia	${I_b}^x x$	12.59 $kg * m^2$
Pitch Moment of Inertia	${I_b}^y y$	12.48 $kg * m^2$
Yaw moment of Inertia	${I_b}^z z$	0.66 $kg * m^2$
Body mass	m_b	51.66 kg
Coloumb Friction torque	D_c	3.82 $N * m$
Viscous Friction coefficient	D_v	3.68 $N * m * s/rad$
Ball drive torque constant	K_i	2.128 $N * m/A$

4 State space representation of the system

In order to simulate the robot in MATLAB, it is necessary to first convert the system into the state space representation. The general form of the state space representation is given by

$$\dot{x} = Ax + Bu; \quad y = Cx + Du; \tag{8}$$

The state variable for the given generalised coordinate $q = \begin{bmatrix}\theta & \phi\end{bmatrix}^T$, where θ is the ball angle, and ϕ is the body angle for the robot, can be chosen as $x = \begin{bmatrix}\theta & \phi & \dot{\theta} & \dot{\phi}\end{bmatrix}^T$

The matrices A, B, C, D are given as follows:

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \frac{\partial\ddot{\theta}}{\partial\theta} & \frac{\partial\ddot{\theta}}{\partial\phi} & \frac{\partial\ddot{\theta}}{\partial\dot{\theta}} & \frac{\partial\ddot{\theta}}{\partial\dot{\phi}} \\ \frac{\partial\ddot{\phi}}{\partial\theta} & \frac{\partial\ddot{\phi}}{\partial\phi} & \frac{\partial\ddot{\phi}}{\partial\dot{\theta}} & \frac{\partial\ddot{\phi}}{\partial\dot{\phi}} \end{bmatrix}; \quad B = \begin{bmatrix} 0 \\ 0 \\ \frac{\partial\ddot{\theta}}{\partial\tau} \\ \frac{\partial\ddot{\phi}}{\partial\tau} \end{bmatrix} \tag{9}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}; D = 0; \tag{10}$$

The expression for $\ddot{\theta}$ is given below:

$$\ddot{\theta} = \frac{\tau(\alpha+\gamma+2\beta\cos\phi)}{(\alpha\gamma-\beta^2\cos\phi^2)} - \frac{(D_v\dot{\theta}+D_c sgn\dot{\theta})(\alpha+\gamma+2\beta\cos\phi)}{(\alpha\gamma-\beta^2\cos\phi^2)} - \frac{(2\beta\sin\phi\dot{\phi}(\alpha+\beta\cos\phi))}{(\alpha\gamma-\beta\dot{\phi}^2\cos\phi^2)} + \frac{(\beta\dot{\phi}^2\sin\phi(\alpha+\gamma+2\beta\cos\phi))}{(\alpha\gamma-\beta^2\cos\phi^2)} - \frac{(\beta\gamma\sin\phi)(\alpha+\beta\cos\phi)}{r(\alpha\gamma-\beta^2\cos\phi^2)}$$

Differentiating the expression above with respect to θ, ϕ, $\dot{\theta}$ and $\dot{\phi}$ and linearizing about the point of unstable equilibrium i.e. all variables are zero, we get

$\frac{\partial\ddot{\theta}}{\partial\theta} = 0$

$\frac{\partial\ddot{\theta}}{\partial\phi} = \frac{\beta\gamma(\alpha+\beta)}{r(\alpha\gamma-\beta^2)}$

$\frac{\partial\ddot{\theta}}{\partial\theta} = 0$

$\frac{\partial\ddot{\theta}}{\partial\dot{\phi}} = 0$

$\frac{\partial\ddot{\theta}}{\partial\tau} = \frac{\alpha+\gamma+2\beta}{\beta^2-\alpha\gamma}$

The expression for $\ddot{\phi}$ is given as follows:

$$\ddot{\phi} = \frac{(\alpha\beta\gamma \sin\phi)}{(r\alpha\gamma - \beta^2 \cos\phi^2)} + \frac{(2\alpha\beta \sin\phi)}{(\alpha\gamma - \beta^2 \cos\phi^2)} - \frac{(2\beta \sin\phi\dot{\phi}(\alpha + \beta\cos\phi))}{(\alpha\gamma - \beta^2 \cos\phi^2)} - \frac{(\tau(\alpha + \beta\cos\phi))}{(\alpha\gamma - \beta^2 \cos\phi^2)} + \frac{(D_v\dot{\theta} + D_c sgn(\ddot{\theta}))(\alpha + \beta\cos\phi)}{\alpha\gamma - \beta^2\cos\phi^2}$$

Differentiating the expression above with respect to θ, ϕ, $\dot{\theta}$ and $\dot{\phi}$ and linearizing about the point of unstable equilibrium i.e. all variables are zero, we get

$\frac{\partial\ddot{\phi}}{\partial\theta} = 0$

$\frac{\partial\ddot{\phi}}{\partial\phi} = \frac{\alpha\beta\gamma}{r(\alpha\gamma-\beta^2)}$

$\frac{\partial\ddot{\phi}}{\partial\dot{\theta}} = 0$

$\frac{\partial\ddot{\phi}}{\partial\dot{\phi}} = 0$

$\frac{\partial\ddot{\phi}}{\partial\tau} = \frac{\alpha+\beta}{\beta^2-\alpha\gamma}$

In the values obtained above, we substitute the ballbot parameters, from Table 1. After this operation, following state space matrices are obtained:

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -6082.8 & 0 & 0 \\ 0 & 92.8 & 0 & 0 \end{bmatrix} B = \begin{bmatrix} 0 \\ 0 \\ 5.0979 \\ -0.4952 \end{bmatrix} C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{11}$$

4.1 Controllability and Observability

The controllability of the system is checked by calculating the controllability matrix, given by $\begin{bmatrix} B & AB & A^2B & A^3B \end{bmatrix}$.

If the above mentioned matrix is full ranked, then the system is said to be controllable. Using the state space representation of the ballbot calculated above, we determine the rank of the controllability matrix. We can use the ctrb function to directly determine the controllability matrix of A and B in MATLAB. Then, from the $rank(ctrb(A, B))$ we can deduce that if the rank is 4 and if it matches the dimensions of the system, then it is controllable. The controllability matrix is

$$\begin{bmatrix} 0 & 5.09 & 0 & 3012.2 \\ 0 & -0.4952 & 0 & -46 \\ 5.0979 & 0 & 3012.2 & 0 \\ -0.4952 & 0 & -46 & 0 \end{bmatrix}$$

In order to check whether the system is observable or not, we calculate the observability matrix. The observability matrix for this system is given by $\begin{bmatrix} C \\ AC \\ A^2C \\ A^3C \end{bmatrix}$

We can use the obsv function to directly determine the observability matrix of A and B in MATLAB. Then, from the $rank(obsv(A, B))$ we can deduce that if the rank is 4 and if it matches the dimensions of the system, then it is observable.

5 Control Architecture

The control architecture plays a crucial part in stabilizing and driving this system. Given the highly dynamic nature of the robot, it is necessary to have a robust control system capable of handling various disturbances.

5.1 Balancing Control

Due to the inherently unstable design of the ballbot, it can tip over and fall down. The function of the balancing controller is to prevent the ballbot from tipping over to large angles. It regulates the roll and pitch angles. The balancing controller is a PID controller. The values of proportional, integral and derivative gain were tuned manually. The controller takes feedback of the body angle from the ballbot block, and compares it with the desired body angle. This error is then fed to the PID block and it is minimized. Figure below shows the block diagram of the controller.

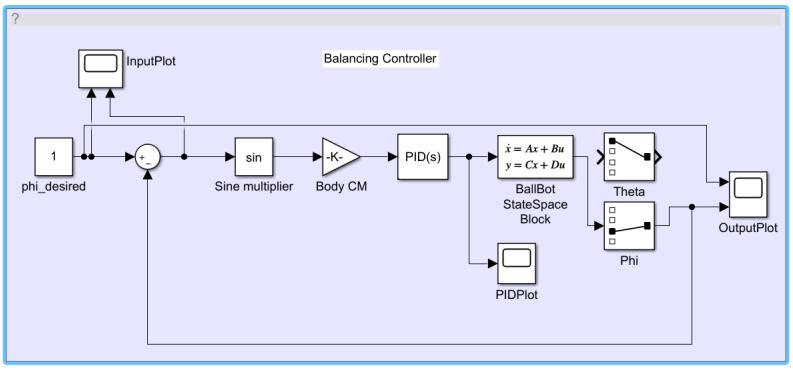

Figure 6: Balancing Controller Block Diagram

The controller was tuned in MATLAB, and the output body angle was found to be within desired limits.However, the balancing controller does not attempt to keep the ballbot at its current position. As a result, it is possible that the robot gets displaced from its home position while performing the balancing operation. This problem is solved by introducing an outer loop, comprising of the stationkeeping controller and velocity controller. A sample output from the balancing control is given below.

5.2 Outer Loop Control

To accomodate for the various trajectories that are possible by the ballbot, there needs to be a controller which can track the desired position as well as the desired velocity. This is done by the following 2 controllers.

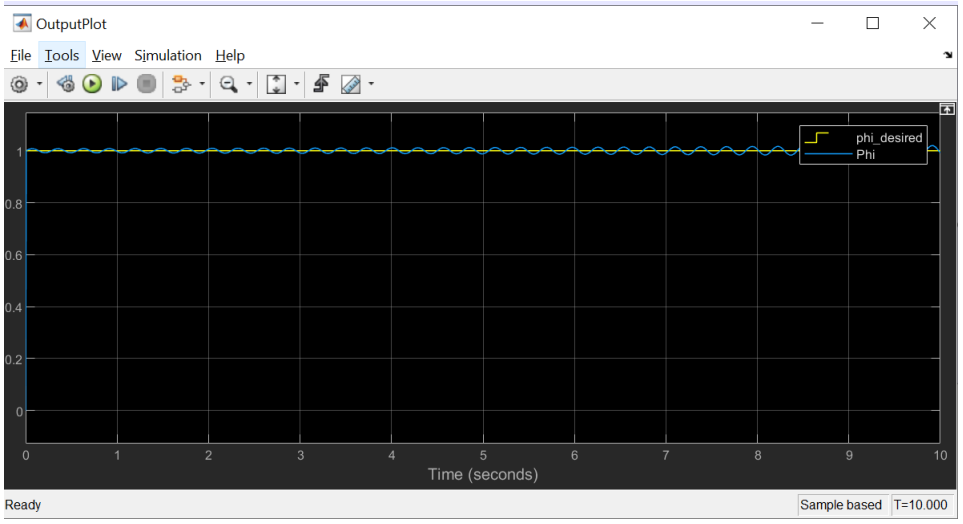

Figure 7: Balancing Controller output

5.2.1 Stationkeeping Control

In the event that the ballbot is pushed, it needs to track back to it's base position. This function is done by the Stationkeeping controller, as shown in figure X. It is a PD controller, which outputs the required body angle. It takes feedback the ball angle, and accordingly adjusts its output. This is the reference signal for the Balancing controller discussed in the previous section. Figure below shows the block diagram of the Station keeping Controller. In order to prevent an aggressive response that might lead to the robot tipping over,

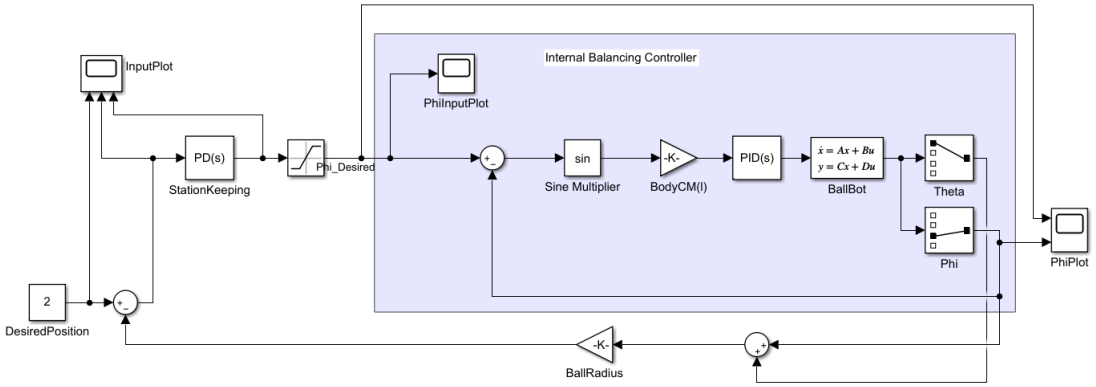

Figure 8: StationKeeping Controller

the output of this controller is limited. This prevents the body angles from reaching dangerously high values. A sample output from the stationkeeping control is shown.

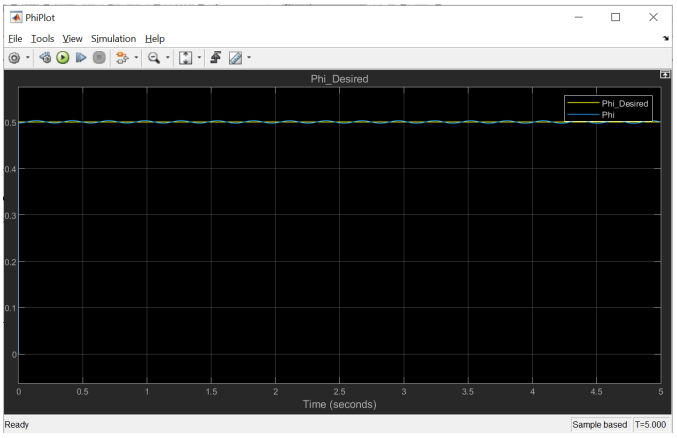

Figure 9: Station Keeping Controller output

5.2.2 Velocity Control

The velocity controller is helpful in 2 scenarios: when the ballbot is being stabilized after a disturbance and when it is being controlled via joystick inputs. It is a PI controller, giving output the desired body angles necessary for optimum velocity. Similar to the stationkeeping controller, its output is limited in order to avoid large body angles. This controller takes input the ball angular velocity, and thereby calculates the required body angles.

5.3 Yaw Control

The purpose of the Yaw Controller is to allow a full 360° rotation of the ball without any change in the orientation of the body. It consists of two loops, an inner PI loop that controls the yaw angle ψ, and an outer PD loop that is used for feedback compensation for both ψ and it's yaw angular velocity ψ'. The Yaw angle is obtained by integrating the yaw angular velocity. In order to avoid high undesired yaw angles, the outer feedback loop is saturated, which also helps the inner PI loop that results in avoiding the use of anti-windup logic. For any yaw angle rotation, the body angles are transformed using an angle offset provided by the absolute yaw encoder.

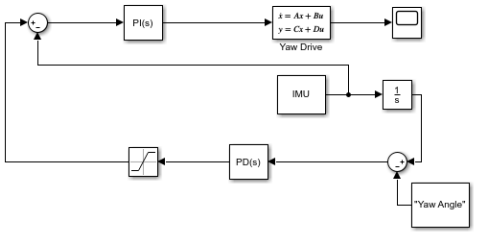

Figure 10: Yaw Controller

5.4 Legs Adjustment Control

The legs adjustment control is used primarily to introduce control on the deployment of legs. It has both legs up control and legs down control for each leg. The legs up controller is a PI control which stops when the velocity becomes less than threshold. The legs down controller has an inner PI control for the leg velocity and an outer PD control for both position and velocity of leg analogous to the controller employed to manage yaw. The legs are not strong enough to support a falling ballbot, but since they are working individually, it gives them an advantage on uneven surfaces. The block diagram for the legs adjustment control is given below.

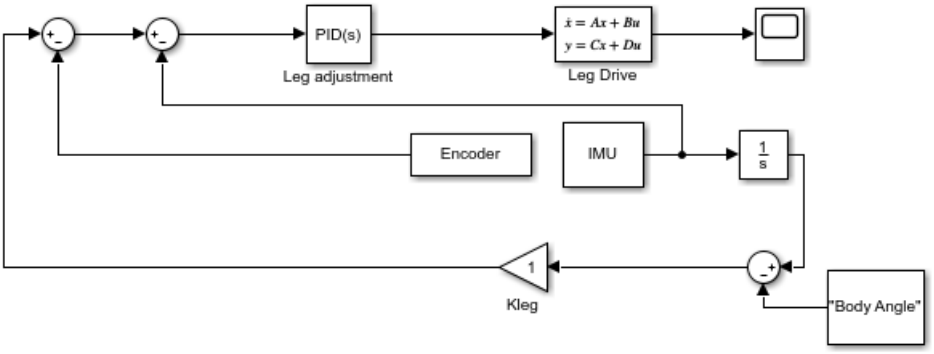

Figure 11: Leg Adjustment Control

There are two states in which the robot operates, a Dynamically Stable State (DSS) where the legs are attached to the body to balance the robot on the ball and a Statically Stable State (SSS) when the legs are deployed on ground making it an over-constrained system. The legs up controller has to work simultaneously with the balancing controller as the body will rely on the ball for which there needs to be balance on the ball. This automatic transition from SSS to DSS can cause undesirable transient if the body angle is large in SSS. Hence, a Legs adjust controller is used to avoid such a scenario. To achieve body angles close to 0, there exists a linear relationship between the leg position ξ and body angle ϕ which is given by, $\xi = K_{leg}\phi + c_{leg}$.

The transition from DSS to SSS is employed when the legs down controller along with balancing controller are in action. As soon as the hoof switch sends a true signal, the balancing controller is shut off and thus, the robot achieves stability on its legs.

6 Trajectory Planning

There is a conflict between the balancing control and stationkeeping control when allowed to track a desired path to overcome the dynamics of the ballbot. This can be avoided by using the balancing control to track the body angle trajectories and hence allowing rest to rest motions on an even surface. The dynamic constraint equation can be determined from the equation of the motion given as,

$$(\alpha + \beta\cos(\phi))\ddot{\theta} + (\alpha + \gamma + 2\beta\cos(\phi_p))\ddot{\phi} - \beta\sin(\phi_p)\dot{\phi}^2 - \frac{\beta g\sin(\phi_p)}{r_w} = 0 \tag{12}$$

The above equation forms a second-order non-holonomic constraint equation that restricts the trajectory to follow a set of configurations only. The trajectory planner uses this dynamic constraint equation to plan the body angle. The equation from above can be re-written as,

$$\ddot{\theta} = f(\phi, \dot{\phi}, \ddot{\phi}) = \frac{\frac{\beta g\sin(\phi)}{r_w} + \beta sin(\phi)\dot{\phi}^2 - (\alpha + \gamma + 2\beta\cos(\phi))\ddot{\phi}}{\alpha + \beta\cos(\phi)} \tag{13}$$

The body angle of the ballbot will be zero to achieve rest-to-rest motion. Using the parametric trajectory from [8] for the body angle, we can write,

$$\phi_p(t) = {\phi_p}^{a1}\sec h(\frac{k*(2t - t_m - t_0)}{t_m - t_0}) + {\phi_p}^{a2}\sec h(\frac{k*(2t - t_f - t_m)}{t_f - t_m}) + {\phi_p}^{0} \tag{14}$$

where ${\phi_p}^{a1}, {\phi_p}^{a2}$ are the amplitudes for hyperbolic secant function with t_f as the final time. This trajectory equation was first defined [7]. Here, $\phi(t_0) = \phi(t_f) = 0$. $t_m = \frac{t_0+t_f}{2}$ is the median time, k is a scalar whose value is chosen optimally to 9. If k is large then the peak is narrower in the graph whereas for a smaller k, it is broader. So, in general, the body angle depends on ${\phi_p}^{a1}, {\phi_p}^{a2}, t_f$ whose values need to be found such that $\theta_p(t)$ reaches ${\theta_f}^{d}$. With these parameters which are sensitive to intial values, the ball angle trajectory can be solved numerically using Nelder-Mead simplex method with $t_0 = 0$, $(\theta(t_0), \dot{\theta(t_0)}) = (\theta_0, 0)$. It is thus an optimisation problem, where we find the parameters ${\phi_p}^{a1}, {\phi_p}^{a2}$ and t_f of the body angle $\phi_p(t)$ such that the objective function

$$J = w_1(\theta(t_f) - \theta_f)^2 + w_2\dot{\theta}^2(t_f) + \int_0^{t_f}(w_3 t + w_4\tau^2)dt \tag{15}$$

has a minimum subject to the dynamic constraint given above. In MATLAB, this can be solved using the $fminsearch$ method. The weights w_1 and w_2 are higher to ensure that $\theta(t_f) = \theta_d^f$ and $\dot{\theta}(t_f) = 0$, whereas the weights w_3 and w_4 find the relative cost between control and time effort.

Using the values for the parameters from [5], we get the following trajectory of the body angle

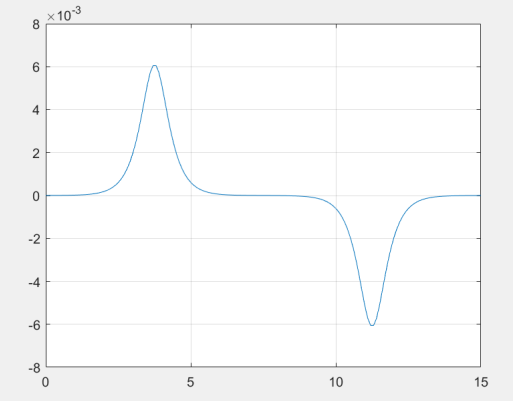

Figure 12: Ball Angle Trajectory

6.1 Feedback trajectory tracking Controller

It is a PID controller with manually tuned gains used to ensure accurate tracking of the ball. Since, open loop control input fails due to modelling errors, nonlinear friction, perturbation, and wrong initial conditions. The input body angle ϕ_p is fed with ball angle θ that produces an error angle ϕ_c. This error angle corrects the body trajectory to reach the desired trajectory $\theta_p(t)$. Internally, the balancing controller tracks the desired body angle trajectory $\phi_d(t)$, which is, $\phi_d(t) = \phi_p(t) + \phi_c(t)$. The feedback term is saturated to avoid undesirable body angles that can drive the system unstable. This controller requires input the desired body angle trajectory, $\phi_p(t)$ and the desired ball angle trajectory.

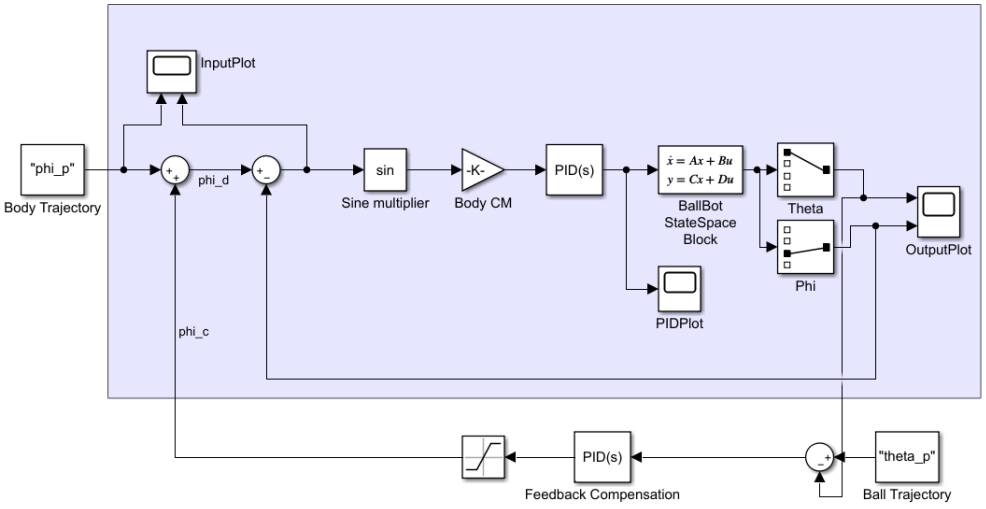

Figure 13: Feedback Trajectory Tracking Controller

7 Discussion

The report talks about the design and control architecture of the Ballbot. It also describes the trajectory plan for the robot to achieve a desired ball motion. With the four wheel inverse mouse-ball drive and a tall narrow structure, the ballbot has a flexibility to move in constricted human terrains. The strong control architecture allows the ballbot to achieve stability in any configuration, from even-uneven surfaces to large disturbances applied directly on the robot. The balancing controller helps the ballbot to stabilise about a given equilibrium point but also helps in maintaining body posture while in motion or in transition from DSS to SSS. The simulation results show that for an input such as the parametric trajectory equation $\phi_p(t)$, the ballbot is able to balance and converges to the input signal, to achieve the desired ball motion.

Further, an LQR controller can be implemented on the ballbot for the balancing control/ station keeping control. It can be used to achieve robust stability with a minimized cost function. Since, it is computationally efficient, the result can be simulated at greater pace and can be implemented on a real system.

8 Conclusion

The dynamic model of the ballbot was computed. The conversion to state space helped us to design the controllers for different actions. The PID control mechanism of balancing control was manually tuned to achieve stability over the given angle. The stationkeeping control mechanism of PI loop was simulated. It stabilises the ball on a given equilibrium point. Yaw control was achieved with the PD control loop and the legs adjust control using both PI and PD loop for each of the actions that included deployment and rollback. The simulation of balancing control also helped in analysing the trajectory for a desired ball motion. A circular trajectory was simulated in V-REP and was compared with the desired trajectory.

9 Drawbacks and Challenges

Although the ballbot is able to execute various trajectories with graceful motion, it has some drawbacks over the traditional multiple contact design robots. The primary challenge is the operation of the robot in case of disturbances. A control system capable of handling such disturbances is necessary. Otherwise, the robot can be potentially hazardous to the humans around it. Multiple sensors and mechanisms must be provided in order to ensure safe operation of the robot.Moreover, the robot cannot be operated on rough terrain. This brings a limitation on the operational domain of the robot. It is important to select the appropriate material for the rollers and the ball. Appropriate material will ensure that they do not slip amongst themselves. The slipping can produce unnecessary vibrations and jerky motion.

References

[1] R. Kelly, J. Llamas, and R. Campa. A measurement procedure for viscous and coulomb friction. *IEEE Transactions on instrumentation and measurement*, 49(4):857–861, 2000.

[2] T. Lauwers, G. Kantor, and R. Hollis. One is enough! In *Proc. Int'l. Symp. for Robotics Research*, pages 12–15, 2005.

[3] T. B. Lauwers, G. A. Kantor, and R. L. Hollis. A dynamically stable single-wheeled mobile robot with inverse mouse-ball drive. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 2884–2889. IEEE, 2006.

[4] U. Nagarajan, G. Kantor, and R. Hollis. The ballbot: An omnidirectional balancing mobile robot. *The International Journal of Robotics Research*, 33(6):917–930, 2014.

[5] U. Nagarajan, G. Kantor, and R. L. Hollis. Trajectory planning and control of an underactuated dynamically stable single spherical wheeled mobile robot. In *2009 IEEE International Conference on Robotics and Automation*, pages 3743–3748. IEEE, 2009.

[6] H. G. Nguyen, J. Morrell, K. D. Mullens, A. B. Burmeister, S. Miles, N. Farrington, K. M. Thomas, and D. W. Gage. Segway robotic mobility platform. In *Mobile Robots XVII*, volume 5609, pages 207–220. International Society for Optics and Photonics, 2004.

[7] J. A. Rosas-Flores, J. Alvarez-Gallegos, and R. Castro-Linares. Control of an underactuated planar 2r manipulator: experimental results. *IFAC Proceedings Volumes*, 35(1):265–270, 2002.

[8] J. A. Rosas-Flors, J. Alvarez-Gallegos, and R. Castro-Linares. Trajectory planning and control of an underactuated planar 2r manipulator. In *Proceedings of the 2001 IEEE International Conference on Control Applications (CCA'01)(Cat. No. 01CH37204)*, pages 548–552. IEEE, 2001.

A Appendix

A.1 Matlab Code

```
clc;
close all;
clear all;

% Initialise the symbols to be used
syms a; % alpha
syms b; % beta
syms g; % gamma
syms phi; % body angle
syms theta; % ball angle
syms phid; % body angular velocity
syms thetad; % ball angular velocity
syms Dc; % coloumb friction coefficient
syms Dv; % viscous friction coefficient
syms r; % ball radius
syms t; % Torque T

%Defining the Mass matrix
M = [a a+b*cosd(phi); a+b*cosd(phi) a+g+2*b*cosd(phi)];
%disp(M);

%Coriolis force Matrix
C = [-b*sind(phi)*phid*phid; -b*sind(phi)*phid*phid];
%disp(C);

%Potential energy Matrix
G = [0;(-b*g*sind(phi))/r];
%disp(G);

%Friction Matrix
D = [Dc*sign(thetad)+Dv*thetad; 0];
%disp(D);

%Torque Matrix
T = [t;0];

%The dynamic equation solving for theta'' and phi''
qdd = mtimes(inv(M),T) - mtimes(inv(M),G) - mtimes(inv(M),C); %-
    mtimes(inv(M),D)
%disp(qdd);
```

```
40
41 %To define the A Matrix
42 dftheta = diff(qdd, theta);
43 %disp(dftheta);
44
45 dfphi = diff(qdd, phi);
46 %disp(dfphi);
47
48 dfthetad = diff(qdd, thetad);
49 %disp(dfthetad);
50
51 dfphid = diff(qdd, phid);
52 %disp(dfphid);
53
54 dfthetatd = diff(qdd, t);
55 %disp(dfthetatd);
56
57 %Calculate the A matrix with the parameters from the table in
       report
58 theta = 0;
59 phi = 0;
60 thetad = 0;
61 phid = 0;
62 % alpha = Iw + (mw+mb)*rw^2
63 a = 0.0174 + (2.44 + 51.66)*(0.1058)*(0.1058);
64 % beta = mb*rw*lb
65 b = 51.66 * 0.1058 * 0.69;
66 % gamma = Ib + mbIb^2
67 g = 12.48 + 51.66 * 0.69 * 0.69;
68 r = 0.1058;
69 N1 = (b * g * (g + b));
70 D1 = ( r * (b * b - a * g));
71 N2 = (a * b * g);
72 D2 = ( r * (-b * b + a * g));
73
74 N3 = -(a + g + 2 * b);
75 D3 = (b * b - a * g);
76 N4 = (a + b);
77 D4 = (b * b - a * g);
78
79 %To define B matrix
80 dfthetadd = N1/D1;
81 dfphidd = N2/D2;
```

```
82
83 dfthetatd1 = N3/D3;
84 dfphitd = N4/D4;
85
86 %Procedure to find A, B, C, D matrices to convert into State
       space
87 A = [0 0 1 0; 0 0 0 1; 0 dfthetadd 0 0; 0 dfphidd 0 0];
88 disp(A);
89 B = [0; 0; dfthetatd1; dfphitd];
90 disp(B);
91 C = eye(4);
92
93 %For the system to be controllable, the rank must be equal to
       matrix A rank
94 if length(A) == rank(ctrb(A, B))
95     disp('The system is Controllable');
96     %disp(rank(ctrb(A, B)));
97 else
98     disp("System is not Controllable");
99 end
100 %For the system to be controllable, the rank must be equal to
       matrix A rank
101 if length(A) == rank(obsv(A, C))
102     disp('The system is Observable');
103     %disp(rank(obsv(A,C)));
104 else
105     disp("System is not Observable");
106 end
```

A.2 V-REP Simulation results

The ballbot was simulated in V-REP environment. Following are the results obtained from the simulation:

1. Circular trajectory tracking

The violet circle denotes the actual path of the robot, whereas the white circle denotes the desired path. As can be seen from the graph, the ballbot tracks the target trajectory closely.

2. Body angles during straight line motion

This graph indicates the body angle beta as the ballbot moves to and fro in a straight line between 2 points. The zero angle is reached when the ballbot is in the middle of its trajectory. The ballbot tilts ahead while accelerating and tilts backward while stopping. This pattern is visible in the body angle graph

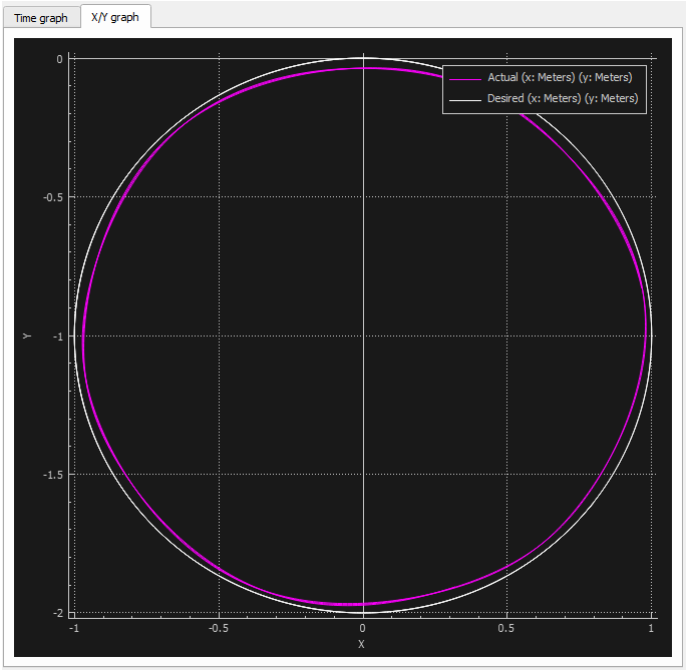

Figure 14: Circular Trajectory Tracking

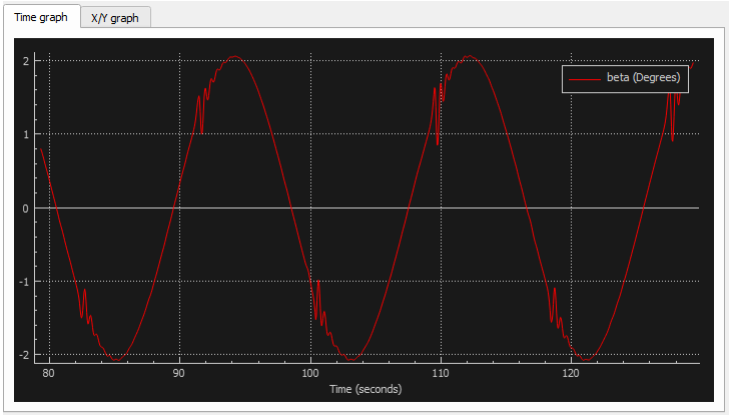

Figure 15: Body angle during straight line motion

3. Velocity profile

This graph indicates the velocity profile of the robot during a straight line motion. There are 2 velocities, one denotes the overall velocity of the body and the other denotes the velocity along the X axis

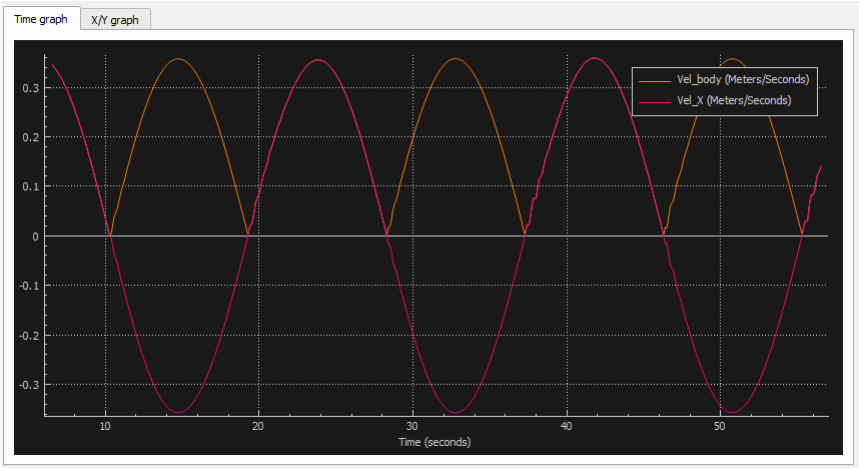

Figure 16: Velocity during straight line motion

A.3 Dynamic Equation Derivation

This section presents the derivation of the dynamic equation. These are derived using Euler Lagrange equations

The Euler Lagrange equation is given by the following generalized equation:

$\mathrm{d}(\partial L/\partial q')/\mathrm{d}t$ - $\partial L/\partial q = \tau$

The term L is given by the difference between the kinetic energy and the potential energy. q is, as mentioned earlier, $\theta\phi$.

$L = KE_{linear}ball + KE_{linear}body + KE_{rotation}ball + KE_{rotation}body - PE_{ball} - PE_{body}$

KE is the kinetic energy, and PE denotes the potential energy.

Consider the motion of the ball

$x_{ball} = r(\theta + \phi)$

$v_{ball} = r(\theta' + \phi')$

$\omega_{ball} = \theta' + \phi'$

$KE_{linear}ball = \frac{1}{2}M_{ball}r^2(\theta' + \phi')^2$

$KE_{rotation}ball = \frac{1}{2}I_{ball}(\theta' + \phi')^2$

$PE_{ball} = 0$

Next, consider the motion of the body

$x_{body} = r(\theta + \phi) + L\sin\phi$

$z_{body} = L\cos\phi$

$v_x body = r(\dot{\theta} + \dot{\phi}) + L\cos\phi\phi'$

$v_z body = -L\sin\phi\phi'$

$KE_{linear} body = \frac{1}{2}M_{body}((r(\theta' + \phi'))^2 + (L\cos\phi)^2 + (L\sin\phi)^2 + 2rL(\theta' + \phi')\cos\phi\phi')$

$KE_{rotation} body = \frac{1}{2}I_{body}(\phi')^2$

$PE_{body} = M_{body}gL\cos\phi$

Therefore, the expression for L is given as follows:

$L = \frac{1}{2}M_{ball}r^2(\theta' + \phi')^2 + \frac{1}{2}I_{ball}(\theta' + \phi')^2 + \frac{1}{2}M_{body}((r(\theta' + \phi'))^2 + (L\cos\phi)^2 + (L\sin\phi)^2 + 2rL(\theta' + \phi')\cos\phi\phi') + \frac{1}{2}I_{body}(\phi')^2 - M_{body}gL\cos\phi$

Differentiating the above expression as per the Euler Lagrange equation, we get the following matrix notation

$$M(q) = \begin{bmatrix} M_{ball}r^2 + M_{body}r^2 + I_{ball} & M_{ball}r^2 + M_{body}r^2 + I_{ball} + M_{body}rL\cos\phi \\ M_{ball}r^2 + M_{body}r^2 + I_{ball} + M_{body}rL\cos\phi & M_{ball}r^2 + M_{body}r^2 + I_{ball} + M_{body}L^2 + I_{body} + 2M_{body}rL\cos\phi \end{bmatrix}$$

$$C(q, \dot{q}) = \begin{bmatrix} -M_{body}rL\sin\phi\dot{\phi}^2 \\ -M_{body}rL\sin\phi\dot{\phi}^2 \end{bmatrix}$$

$$G(q, \dot{q}) = \begin{bmatrix} 0 \\ -M_{body}gL\sin\phi \end{bmatrix}$$

The friction force is given by $D_v\dot{\theta} + D_c sgn(\dot{\theta})$, where D_v and D_c are the viscous damping parameter and coulomb parameter respectively. These equations are obtained from [1].

Therefore, the frictional torque vector can be written as

$$D(\dot{q}) = \begin{bmatrix} D_v\dot{\theta} + D_c sgn(\dot{\theta}) \\ 0 \end{bmatrix}$$