Closed-loop Position Control of an Automated Treadmill

Oles Synyutka
Department of Mechanical Engineering
University of Michigan
Ann Arbor, Michigan 48103
Email: oles@umich.edu

Abstract—The Robotics and Motion Laboratory wishes to test a bipedal robot by having it walk continuously on a treadmill. The aim of this research was to design a control system that regulates the position of the robot as it walks. Pulsations of the treadmill, which were caused by low frequency motor operation, were eliminated through installation of a speed reducer. Oscillations in velocity measurements were reduced by realigning the encoder body to the treadmill roller. Next, an open-loop system was implemented that allows researchers to easily set a desired treadmill velocity. A closed-loop position control system was then designed and simulated. The system uses estimates of robot velocity and measurements of position to keep the robot relatively centered on the treadmill. Additionally, the system ignores small errors in position, which limits the treadmill’s acceleration and prevents the dynamics of the robot from being affected.

I. INTRODUCTION

A. Goals of Research

The Robotics and Motion Laboratory specializes in the field of legged robotics, and is currently developing a bipedal robot. A few years ago, a custom treadmill was manufactured so that the robot could walk continuously, and so researchers could obtain valuable information about its performance. An obvious requirement for running these tests is that the robot must not fall off the edges of the treadmill. The researchers also wanted the convenience of not having to manually adjust the treadmill velocity to maintain the robot’s position. Thus, the main goal of this research project was to design a closed-loop control system that would keep the robot reasonably centered on the treadmill as it walks with varying velocity.

A previous graduate student had conducted an ME 590 project where he worked on developing such a system [1]. In his findings, he reported that the treadmill speed had significant oscillations at steady-state, but he did not have time to determine the reason or fix the issue. Reducing these unwanted oscillations was a key objective in this research. In addition, the student was able to achieve good tracking of the desired treadmill velocity, but did not address the positioning of the robot on the treadmill. Therefore, a position control system still needed to be implemented.

Although it is important that the robot does not come too close to the edges of the treadmill, it is also important that the treadmill has relatively low accelerations. The purpose of the treadmill is to simulate the robot walking on the ground (at a relatively constant speed), and if the treadmill accelerates too much, it will impact the dynamics of the robot’s walk. Obviously, if the robot suddenly approaches the edge of the treadmill, the system will need to accelerate quickly to recenter the robot. In that extreme case, there is no way of achieving low acceleration. However, once the robot is within a safe distance from the center of the treadmill, high accelerations are unnecessary. Hence, we wanted our control system to disregard small deviations of the robot from the center of the treadmill, and have high acceleration only if the position exceeded a specified threshold.

B. Overview of Treadmill System

An image of the treadmill system is shown in Fig. 1. The main treadmill assembly connects to a two-pulley belt drive (transmission ratio of $n_{12} = 2$), which is powered by an induction motor. The induction motor is controlled with a variable frequency drive (VFD). Above the treadmill is the bipedal robot RAMone, which is restricted to 2D motion via a planarizer. Note that in Fig. 1, there is a speed reducer (transmission ratio of $n_{r1} = 5$) between the motor and belt drive; this reducer was not added until later in the research.

There are two sensors in the treadmill system: a rotary incremental encoder to measure the velocity of the treadmill roller (from which any other velocities can be calculated), and a linear incremental encoder to measure the position of the robot. Data from the sensors is transmitted through an EtherCAT coupler, which connects to a target PC. The target PC is connected to a host PC running MATLAB and Simulink. Using xPC Target, we are able to develop models in Simulink on the host PC, upload them to the target PC, and run them in real-time. This allows us to collect sensor data and test out various control systems with ease.

II. OVERVIEW OF INDUCTION MOTOR THEORY

Before discussing the results of our research, we would first like to present a brief overview of induction motors. As implied by their name, these motors work on the principle of electromagnetic induction. An induction motor consists of two main components, the stator and rotor. The stator is composed of multiple conductive windings. To power the motor, a 3-phase alternating current is applied to these windings. The current generates a magnetic field inside the stator, according to Ampere’s Law. Because of the geometry of the windings,
and because the three phases of current are 120° apart, the magnetic field rotates at a constant velocity known as the synchronous speed, \( \omega_s \). This speed is given by

\[
\omega_s = \frac{4\pi f_s}{p}
\]  

(1)

where \( f_s \) is the frequency of the AC current, and \( p \) is the number of poles in the motor (a pole is simply how many windings there are for each phase of current). For a 2-pole motor, such as the one used in our research, the synchronous speed is \( \omega_s = 2\pi f_s \). [2]

As the constant magnetic field in the stator rotates, the magnetic flux inside the rotor varies, inducing a current in the rotor (by Faraday’s Law). Since the current-carrying rotor is inside a magnetic field, a force is produced, resulting in the motor torque \( T_m \). For torque to be produced, there must be a difference between the synchronous speed and the rotor velocity \( \omega_r \). The reason is simple: if there is no relative velocity between the magnetic field and the rotor, then the magnetic flux in the rotor does not vary, so there is zero induced current and hence zero torque. The slip of an induction motor, \( s \), describes the difference between the rotor velocity and synchronous speed, and is given by [2]:

\[
s = \frac{\omega_s - \omega_r}{\omega_s}.
\]  

(2)

Fig. 2 shows a typical torque-speed curve for an induction motor at some arbitrary \( f_s \), and loaded with a torque \( T_L \). If the motor is turned on from rest, it will begin at the starting torque \( T_{st} \), and accelerate through the unstable region to the left of the dashed line. After going past the breakdown torque \( T_{bd} \), the motor is in its stable operating zone. It will reach a steady-state velocity when \( T_m = T_L \); this is known as the motor’s operating point. Note that when \( \omega_r = \omega_s \) (zero slip), no torque is produced. Even when unloaded, motors will never operate exactly at synchronous speed due to friction inside the motor. [3]

The velocity of an induction motor is often controlled by changing the input frequency \( f_s \) (effectively changing the synchronous speed). As the input frequency lowers, the entire torque-speed curve shifts to the left, as shown in Fig. 3. Assuming the load stays the same, the motor will now operate at a lower velocity. In most cases, open-loop control is sufficient. A typical induction motor operating at its rated torque will have less than 3% slip, so the motor runs close to synchronous speed even under loaded conditions [4]. So given a desired motor velocity, the user can determine the necessary input frequency with Eq. 1.

For reasons outside the scope of this report, the ratio between the power supply voltage and input frequency (denoted as V/Hz ratio) must remain constant in order for the motor to maintain the same torque capacity. A variable frequency drive (VFD) is a convenient way of changing input frequency while automatically maintaining the V/Hz ratio. However, as shown in Fig. 3, even if the V/Hz ratio is maintained, the torque capacity of the motor decreases dramatically at low enough frequencies. In other words, the curve shifts to the left but is also scaled down vertically. The reason is that the voltage drop across the stator windings becomes significant compared to the the input voltage. Therefore, running induction motors at very low speeds is generally not a good idea, especially because of overheating issues. [3]
III. PRELIMINARY ANALYSIS OF TREADMILL SYSTEM

Before attempting to implement any open-loop or closed-loop control of the system, we first set out to reduce the oscillations that were reported in previous studies. There was not much documentation regarding the characteristics of the oscillations, so we ran the treadmill system and collected our own data. The input frequency $f_s$ was varied by giving the VFD a 0-10 V reference input (with a potentiometer and power supply), which it converted to a 0-60 Hz frequency. We first ran the treadmill at velocities that were reasonable for the bipedal robot (around 0.75 m/s). Assuming that the motor operates at synchronous speed, these treadmill velocities corresponded to input frequencies of around 5 Hz. These frequencies were extremely low, since the motor was designed to run at around 60 Hz.

At these low frequencies ($< 6$ Hz), the treadmill system produced a very odd behavior, where it visibly sped up and slowed down in a pulse-like manner. At around 1 or 2 Hz, the treadmill sometimes came to a complete stop and then sped up once again. Unfortunately, no data was recorded at these low frequencies. When the input frequency was increased, the pulsing behavior was no longer visible to the eye, but the velocity of the system still appeared to oscillate at steady-state. We were unable to test higher than 20 Hz because we feared that the treadmill components would be damaged.

As stated in Sec. II, when induction motors operate at low input frequencies, their torque capacity drops and they may operate erratically. We suspected that this was causing the system’s odd behavior, and consulted with the manufacturer of the motor. The manufacturer agreed that the motor will not perform properly at those frequencies, and recommended to operate at no less than 30 Hz. However, a 30 Hz input would lead to very high treadmill velocities, so we needed a way to operate the motor fast yet have the treadmill going at a reasonable speed.

It is important to note that at this stage of the research, we believed that the visible pulsations at low frequency and the oscillations at higher frequencies (8-20 Hz) were the same problem. We assumed that the oscillations at higher frequency were simply less noticeable because the motor was spinning at higher speed. In the next section, we will discuss that this assumption was incorrect, and that there were two independent issues going on.

IV. ADDITION OF SPEED REDUCER

To run the motor at high operating frequencies, while having the treadmill go at low speeds, we placed a speed reducer between the motor and belt drive, as shown in Fig. 1. A schematic of the final treadmill system is shown in Fig. 5. The linear treadmill velocity, $v_t$, can be found as follows:

$$v_t = \frac{\omega_3 R_3}{n}$$

where $\omega_3$ and $R_3$ are the velocity and radius of the treadmill roller (respectively), and $n$ is the total transmission ratio between the motor and roller.

Identical tests were run on the new system, and the oscillations were still present, as shown in Fig. 6. In fact, the oscillations occurred even at the motor’s “designed” frequency of 60 Hz, so clearly they were not being caused by low frequency operation.
V. REALIGNMENT OF INCREMENTAL ENCODER

To obtain a better understanding of the oscillatory behavior, the Discrete Fourier Transforms (DFT) of the velocity profiles were analyzed. As expected, the plots showed spikes at frequencies other than 0 Hz. To determine the cause, these frequencies were compared to all of the rotational velocities in the system. It was found that the frequency of oscillation (in rad/s) was nearly identical to the velocity of the treadmill roller, \( \omega_1 \), as illustrated in Fig. 7. This indicated that there was some sort of misalignment or eccentricity in the roller’s rotation. While inspecting the roller, we noticed that the encoder shaft was not rotating about its axis. We suspected that the shaft was not screwed into the center of the roller. Depending on how the encoder shaft was positioned relative to the encoder body (which contains the photodiode), the variation in detected velocity would change. The body of the encoder was adjusted, and the oscillations were reduced significantly, as shown in Fig. 8. A comparison between DFT’s before and after encoder realignment is shown in Fig. 9 for \( f_s = 60 \text{ Hz} \). After realignment, the frequency spikes were still present, but at negligible amplitudes. Note that there were additional minor frequencies present, but they were not significant enough to affect system performance. Most induction motor systems have these frequencies because of mechanical imperfections like rotor imbalance or misaligned shaft couplings [5].

After discovering that the steady-state oscillations were a result of encoder misalignment, not a physical issue with the system, we realized that the visible pulsating at low frequencies was a totally separate issue. The motor most likely having a difficult time overcoming the friction in the system, since its ability to produce torque was hindered at those low frequencies. Now that the motor was running at higher frequencies with the speed reducer, the pulsating was no longer an issue. Out of curiosity, we tested the motor at the same low frequencies (1-2 Hz), and the pulsating no longer occurred, most likely because the motor could effectively produce five times more torque than before. As a result, the treadmill could now operate at a much larger range of speeds.

VI. IMPLEMENTATION OF OPEN-LOOP CONTROL

After solving the treadmill’s oscillation problems, the next step was to implement an open-loop system for controlling the velocity of the treadmill. The researchers at the lab wanted the ability to quickly and precisely set the treadmill velocity directly from a PC rather than manually tune a potentiometer.

To perform open-loop control, we first needed to figure out how the motor and treadmill system responded to a given input frequency. With that information, we could then take a desired treadmill velocity and determine its corresponding input frequency. Looking at the velocity profiles of the motor in Fig. 8, we found that when no load was applied (apart
Fig. 8. Upon realignment of the encoder body, the amplitude of the oscillations decreased significantly, indicating that the steady-state oscillations were not an actual physical issue with the system. Nevertheless, the speed reducer was still necessary to prevent pulsations at low frequency.

Next, experiments were conducted where an 86 kg human (eight times heavier than the robot) walked on the treadmill after it reached steady-state. These tests were done at input frequencies ranging from 10-40 Hz. The velocity profiles of the motor showed no difference before and after the person stepped onto the treadmill. The plots are not shown here, since they are essentially identical to Fig. 8. It was not surprising that loading the treadmill had no effect on the motor velocity. In the operating region, the torque-speed curve of an induction motor tends to be very steep, so even at rated torque, a motor runs close to synchronous speed. Moreover, after adding in the 5:1 speed reducer, our motor’s torque-speed curve effectively became five times steeper, which reduced the effect of external loads even further.

Since the induction motor operated at synchronous speed at all reasonable loading conditions, designing an open-loop system was a trivial task. Given some desired treadmill velocity $v_{des}$, the required input frequency can be calculated using Eq. 1 and Eq. 3:

$$f_s = \frac{2\pi R_3}{n} v_{des}$$  \hspace{1cm} (4)

Fig. 10 shows a block diagram of the open-loop system. A rate limiter was applied to $f_s$ so that it would always ramp up gradually to the desired value. This was necessary because increasing the input frequency too quickly would cause the power output of the motor to exceed the rating of the speed reducer, which would result in mechanical damage.

This control system was implemented on the physical treadmill using Simulink and xPC Target, and performed exceptionally well. Fig. 11 shows an example of the system responding to a desired treadmill velocity of $v_{des} = 1.92 \text{ m/s}$. The system shows nearly perfect tracking of the desired velocity. The excellent (and seemingly immediate) tracking would later simplify the design of a position control system.

VII. SIMULATION OF CLOSED-LOOP CONTROL

The final step of this research was to develop a closed-loop control system that would keep the robot from falling...
off the edges of the treadmill. As stated in Sec. I, the main requirements for the system were as follows:

1) The robot must stay relatively centered on the treadmill.
2) The treadmill should have large accelerations only if the robot is at an “unsafe” distance away from center. Otherwise, the treadmill should have very little acceleration, while still guiding the robot back to the center.

The second requirement exists because if the treadmill is constantly accelerating, it will impact the dynamics of the robot’s walk. Therefore, if the robot is close enough to the center of the treadmill, the control system should not take the error in position too seriously, and the treadmill velocity should stay relatively unchanged. However, once the robot strays far enough, the system should quickly return the robot back to the treadmill’s “safe” region.

We will now provide a derivation of the closed-loop control system, and will be referring to Fig. 5. From previous findings, we knew that the motor velocity tracks the synchronous velocity nearly perfectly (given that the frequency is ramped up gradually), so the transfer function for the treadmill speed with respect to input frequency can be written simply as

\[
\frac{V_i(s)}{F_s(s)} = \frac{2\pi R_3}{n} \tag{5}
\]

Next, we define \(x\) as the horizontal position of the robot and \(v_{rob}\) as the robot velocity relative to the treadmill belt (this is the velocity that the treadmill is meant to simulate). We can calculate the Laplace transform of \(x\) as follows:

\[
x = v_{rob} - v_t \tag{6}
\]

\[
X(s) = \frac{1}{s} [V_{rob}(s) - V_i(s)] \tag{7}
\]

Since there is a linear encoder measuring robot position, we can construct a feedback loop where \(x\) is compared to the desired position, \(x_{des} = 0\) (we define zero as the center of the treadmill). The error, \(e = x - x_{des}\), is fed through a PI controller, \(C(s) = K_P + K_I/s\), which outputs the input frequency, \(f_x\). Using Eq. 5 and Eq. 7, we can construct the black part of the block diagram shown in Fig 12. Note that we saturate both the value and derivative of the input frequency, because mechanical damage could occur if the frequency goes above 70 Hz, or if it increases too sharply. Since the frequency is rate-limited, the use of Eq. 5 is well-justified.

In theory, this preliminary feedback system is enough to control the position of the robot. The robot velocity is essentially a disturbance that causes the robot to go off center, and by feeding back the position, the system can reject this disturbance. However, we can further improve the system’s performance because in practice, we will have some idea of what \(v_{rob}\) is. By using an estimate of \(v_{rob}\), we can reduce the effect of the disturbance; this disturbance compensation scheme is shown in the blue portion of Fig. 12. The estimate of robot velocity, \(v_{est}\), can be attained in multiple ways, but the important thing is that the estimate should essentially be a constant. Otherwise, the motor velocity would constantly change with any slight acceleration of the robot. One estimate we could use is the velocity that the robot is commanded to walk at (the researchers will have this value). Alternatively, since we have an encoder on the treadmill roller, we could use Eq. 6 to constantly calculate the robot velocity, and filter out any fluctuations to achieve a relatively constant value.

Regardless of how we estimate \(v_{rob}\), the disturbance will never be fully compensated for. One reason is that Block A and Block B will most certainly not be perfect inverses of each other (this is necessary for perfect compensation [6]). Moreover, the estimate of robot velocity will not be exact, especially if we are choosing it to be constant. To account for these inaccuracies, we include \(e_r\) in our block diagram, which is the relative error of our estimate of \(v_{rob}\).

The final step in our design is to include a pseudo-deadzone for the error signal that is fed into the PI controller. First, we define a threshold position error, \(e_{th}\), that describes the area of the treadmill that we consider to be a safe distance from the center. If the robot is within this threshold \(\{ |e| \leq e_{th}\}\), then we wish for the error to be reduced before entering the controller. This will cause the system to respond slowly to small position errors. If the robot is outside the threshold, the system will respond quickly to get the robot back inside the safe region. The equation for the deadzone is:

\[
\hat{e} = \begin{cases} 
Ae & |e| \leq e_{th} \\
E + (A - 1)e_{th} & e > e_{th} \\
E + (1 - A)e_{th} & e < e_{th} 
\end{cases} \tag{8}
\]

where \(\hat{e}\) is the modified error and \(A < 1\).

Ignoring the nonlinearities in the system, we can show that the disturbance transfer function is

\[
\frac{X(s)}{V_{rob}(s)} = \frac{ne_{r}s}{ns^2 + 2\pi R_3 K_P s + 2\pi R_3 K_I} \tag{9}
\]

Assuming a step input in robot velocity, we can use the final value theorem to calculate the steady-state position, \(x_{ss}\):

\[
x_{ss} = \lim_{s\to0} sX(s) = 0 \tag{10}
\]

Hence, the system has no steady-state error, which makes sense because integral control is being used. Note that if our estimate of robot velocity improves, \(e_r\) becomes lower, and the effect of the robot’s velocity is reduced. Further analysis could be done on the system in Eq. 9, but because of the nonlinearities in the actual system, the results would likely be very different, so further analysis was not worthwhile.

Unfortunately, the robot was malfunctioning toward the end of this research, so the closed-loop system could not be implemented on the physical treadmill. Instead, the system was simulated to verify its performance. Fig. 13 shows the system response to a step in robot velocity; the parameters we used were \(e_{th} = 0.25\ m\), \(e_r = 0.25\), and \(A = 0.001\). The system responds relatively quickly to get the robot back within 0.25 m of center, and once it gets there, the robot very slowly approaches the center of the treadmill, such that the treadmill’s acceleration is small. In this simulation, the robot reaches more than 1 m away from the center, which is not acceptable for our purposes. However, this deviation from
Fig. 12. Block diagram of the closed-loop position control system. The system uses feedback measurements of robot position and an estimate of robot velocity. The dead zone applied to the error signal allows the treadmill to have low acceleration when the robot is close enough to the center of the treadmill.

Fig. 13. Given a step in robot velocity, the robot quickly returns to the "safe" region (between the blue lines), and then slowly approaches the center of the treadmill. Treadmill acceleration is nearly zero during the slow approach.

center is purely dependent on the rate-limitation of the input frequency, and how quickly the robot accelerates to its final velocity. In practice, it will be necessary to gradually ramp up \( v_{rob} \) or allow \( f_s \) to increase more quickly. The purpose of the simulation was simply to demonstrate that the control system works as we intended.

In reality, the robot velocity will have minor fluctuations around some constant value. Fig. 14 shows how the system responds when \( v_{rob} \) oscillates, but with a low enough amplitude that the robot does not leave the safe region of the treadmill. Once the robot returns to within 0.25 m of center, the acceleration of the treadmill is zero even though the robot is constantly deviating away from center. Without the error deadzone, the treadmill velocity would always be changing to compensate for small errors in position. Therefore, the deadzone is necessary if we want the robot’s dynamics to be unaffected by the treadmill.

VIII. CONCLUSION

The goal of this research was to design a control system that will keep a legged robot centered on a treadmill as it walks with varying velocity.

In the first phase of the research, we observed visible pulsations of the treadmill velocity when the induction motor ran at low input frequencies; we also observed steady-state oscillations in velocity for all input frequencies. The visible
pulsations occurred most likely because the torque capacity of induction motors drops significantly at low input frequency. This problem was remedied by the addition of a 5:1 speed reducer, which allowed us to run the induction motor at much higher frequencies. The steady-state oscillations were found to be a result of a poorly centered encoder shaft and misaligned encoder body. Once the encoder body was realigned, the oscillations in our measurements were substantially reduced.

Next, an open-loop system for controlling treadmill velocity was implemented. The system allows the researchers to input a desired velocity, and the treadmill will slowly ramp up to that value. Due to the motor’s steep torque-speed curve (made even steeper by the speed reducer), the system tracks the desired velocity perfectly under any reasonable loading conditions.

Finally, a closed-loop position control system was developed. Through simulation, we showed that the system quickly returns the robot to a safe region of the treadmill, after which the robot slowly approaches the exact center. Even if the velocity of the robot fluctuates, as long as the robot remains in a safe region, the treadmill will show little to no acceleration. This allows for the dynamics of the robot to be unaffected, so that the treadmill simulates real-life walking more accurately.

Once the robot is fixed, we will immediately begin testing the closed-loop system. We will see how the performance compares to our simulation results, and adjust our model accordingly.

ACKNOWLEDGMENT

We would like to thank Nils Smit-Anseeuw for his guidance throughout this entire project, and Rodney Gleason for his assistance with hardware and xPC Target.

REFERENCES