Failure Detection of a Bipedal Robot and Control of an External Active Recovery System

Kevin J. Green
Robotics and Motion Laboratory
Department of Mechanical Engineering
University of Michigan
Email: kevingre@umich.edu

December 18, 2015

Abstract

It is common in legged robotics experiments for the robot to fall. Physical robots are a sizable investment of both time and money, thus it is crucial that when the robot falls it is not damaged. Thus, it is necessary to identify when failure occurs and when to start to take steps to protect the robot. In the RAMLab (Robotics and Motion Laboratory), a high power winch and cable system is used to lift the robot into the air when it fails. This has the advantage that it does not affect the robots dynamics when walking normally. My project is the design and implementation of the control system for this virtual safety net. Required parts of this control system are failure detection, constrained trajectory control, minimal impact tracking, and constant force control. The winch system is heavily constrained due to its ability to only apply force in one direction. This constraint adds complexity to the trajectory control problem design the controller specifically to avoid overshoot. The different control systems were implemented both in simulation and in hardware to characterize behavior to a variety of inputs and develop an accurate model of the system.

1 Motivation

Early robotic gaits were very conservative. They focus on stability and robustness over speed or efficiency. For example Honda's ASIMO walked in a completely statically stable gait. If it froze its joints at any point in time it would no fall over. As a result of this, it had a very high cost of transport and is only able to operate for 15 minutes at a time [1]. As both hardware, controller design and computing power improve, more complex controllers become feasible. One goal of the RAMLab is to explore energy optimality in gaits. Real hardware energy optimal gaits can only be found using hardware experiments. This is because there are many different ways energy is lost in a real system that is not accounted for in simulation. [2] When testing gaits on hardware it is virtually guaranteed that the optimizer will test gaits that are unstable. To accommodate this, a system that mitigates this failure and allows automatic resetting is required.

2 Failure Detection

For a failure mitigation system to property function, it needs to be automatically triggered when the robot fails. We have several different methods of failure detection. Early in the walking gait development cycle we saw the need for software actuator limits. These software limits prevent the joints from over extending. If a joint did over extend, it contacted another part of the robot (usually the main body or a chain). Once we implemented these limits we noticed that the robot rarely fell without one of the joints extending past a limit. We suspect that this results from our use of virtual model control. We decided that this would be a good trigger for the catch system. If the robot passes an actuator limit, it sends the entire robot into a failure state where the motors positions are locked. This indicates that the robot has totally failed and we want to lift it clear of danger.

This covers many cases, but we want to be sure that all failures will be accounted for. We looked at what aspect of failure we wanted to avoid. We want to avoid the robot from damaging itself. Software limits prevent the legs and feet from hitting the body. The other way it could damage itself is through ground contact. The forms of ground contact that we anticipate is knee contact and main body contact. We define several points on the robot and if any of them get too close to the ground we will trigger the catching routine. We have one point on each knee and one at each top corner of the main body. These ground contact conditions are showing in figure 1.

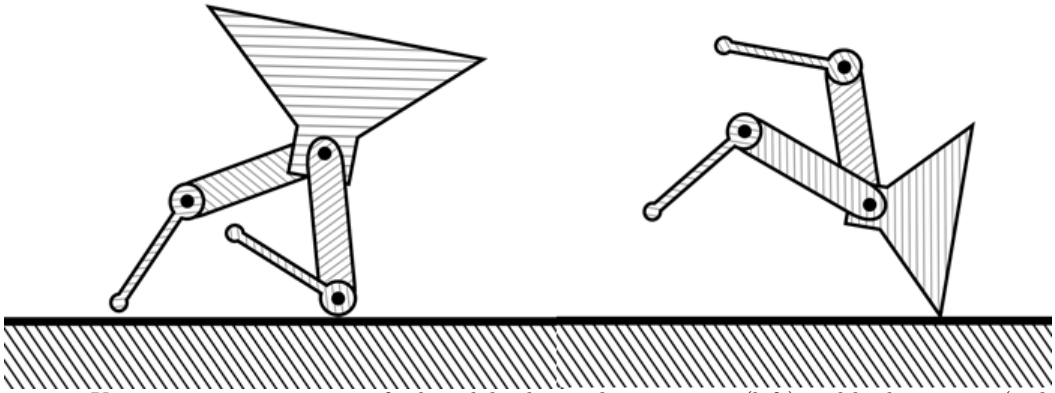

Figure 1: Kinematic representation of a bipedal robot in knee contact (left) and body contact (right)

The reaction to either failure triggers is the same. It will tell the catch system to lift the robot to a set safe height.

3 The Spool System

The physical spool system was manufactured previously. It consists of a high power brushless motor that turns a spool through a chain drive. This spool winds and unwinds thin steel cable which pulls the robot up and down. The set of pulleys on the top of the cart allow the robot to move left and right without any force from the system. On the right side of the system is a hanging mass. The hanging mass allows the system to keeps its line taunt while applying a constant predictable force on the robot. When we want to lift the robot in the air we can wind up the cable until the mass is pressed against the hard stop. When the mass is fully lifted, the cable tension compresses the die spring at the top of the right side. A schematic of the system is shown below in figure 2.

3.1 Spool System Modeling

This is a relatively complicated system. We developed a model that remove the implementation details that are unimportant for the dynamics. The spool motor has a low level controller that controls the motor to follow the velocity commands which are sent to it. If we operate the motor within its velocity, torque and acceleration limits, it is reasonable to separate the motor from the dynamics of the system and treat the motor as a controllable velocity. Further we can remove the complexity of the cable system and the hanging mass and simplify to a one dimensional system. We lump the mass of the robot in with the hanging mass. By doing this we are separating the cable tension into two forces, one from the hanging mass and one from the spring deflection and spring damping. The resulting model is shown in figure 3 and the equation of motion is

$$M_r \ddot{y}_r = F_{sp} + F_{ext} - F_g$$

Where F_g is the gravitational force, F_{ext} is the ground reaction forces on the robot's feet and F_{sp} is the elastic and damping forces in the spring.

We define the spring be single acting. This represents the fact that the system does not add extra force beyond the weight of the hanging mass when the hanging mass is not up against the stop. We first attempt to represent the single acting spring force as,

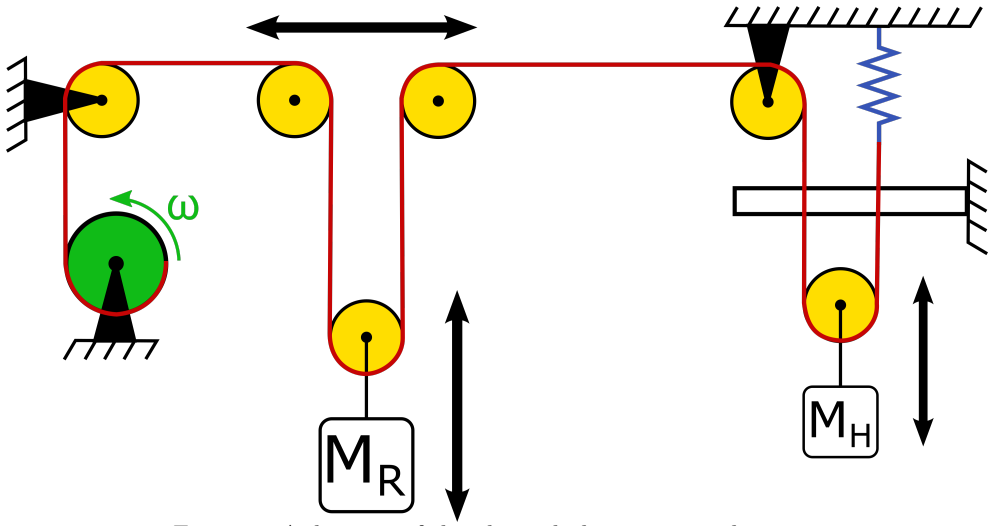

Figure 2: A diagram of the physical planarizer spool system

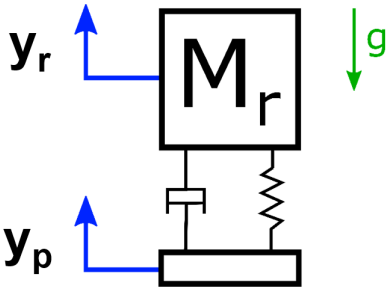

Figure 3: A diagram of the model for the planarizer spool system

$$F_{sp}^{(1)} = \begin{cases} k(y_p - y_r) + b(\dot{y}_p - \dot{y}_r) & y_p \geq y_r \\ 0 & y_p < y_r \end{cases}$$

The majority of damping that is present in the system is part of the spring itself, which is why the damping is also only single acting. A small problem that arises is that this spring force is discontinuous in position. If the velocity is non zero and it crosses zero deflection we get an immediate force. This is problematic both theoretically and for modeling. It is well known that this kind of discontinuous force can result in jitter in simulations[3]. To correct this we redefine our spring force to include a third case. We add a transition zone in between our idea damping and our zero damping. This transition zone damping is a cubic which has continuity and differentiability at the transition points. The final spring force is represented by

$$F_{sp} = \begin{cases} k(y_p - y_r) + b(\dot{y}_p - \dot{y}_r) & y_p > y_r + \epsilon \\ k(y_p - y_r) + b(-2(\frac{y_p - y_r}{\epsilon})^3 + 3(\frac{y_p - y_r}{\epsilon})^2)(\dot{y}_p - \dot{y}_r) & y_r \leq y_p \leq y_r + \epsilon \\ 0 & y_p < y_r \end{cases}$$

We are most interested in this dynamic system when we are controlling the position of the robot. If we are in the range where the spring is applying a zero force, the planarizer cannot apply any force to the robot and thus cannot control the position. We want to avoid that region while lifting the robot. When we are lifting the robot we can assume that the legs are off the ground and that $F_{ext} = 0$. We restrict our system to only operate in the linear region $y_c > y_p$. Omitting the derivation (see appendix A), we can find the following relation:

$$\ddot{y}_r > -g + \frac{B}{M_r}$$
$$\ddot{y}_r > -5.11 \frac{m}{sec^2}$$
$$B \geq |b(\dot{y}_p - \dot{y}_r)|$$

In this equation B is an upper bound on the damping force. The damping was treated as a disturbance to simplify the analysis. We can see that we will stay in our linear region as long as we keep our robot acceleration below a a constant. We will use this fact when we are designing our controller. We will use $B = 47N$, the reasoning can be found in Appendix A.

3.2 Spool System Following Control

The goal of the following controller is to stay close to the robot in case it fails while not affecting the dynamics of robot. The controller we used is a P controllers with a velocity feed forward. The desired position is the robot's current position minus a small offset. The feed forward term is the robot's velocity. The feed forward should give us very good tracking, which is the key to maintaining a constant offset. The equation of the controller is shown below. $\dot{y}_{motor}$ is the commanded planarizer velocity and d is the desired offset between the robot and planarizer.

$$\dot{y}_{motor} = k_p(y_r - d - y_p) + k_d(\dot{y}_r - \dot{y}_p) + k_{\text{ff}}\dot{y}_r$$

Here we see data from the implemented controller. The robot was rapidly moved up and down by a person and the reaction of the planarizer is shown. Figure 4 is of the position of the robot and the planarizer. Figure 5 shows the offset between the robot and the planarizer.

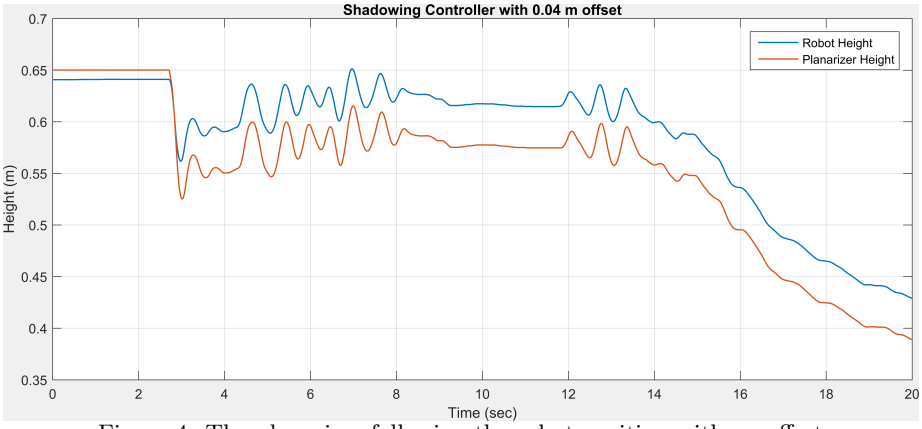

Figure 4: The planarizer following the robot position with an offset

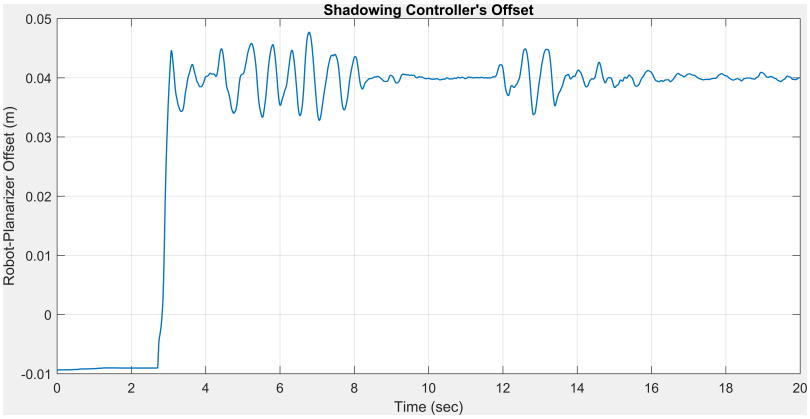

Figure 5: The offset between the robot and the planarizer

3.3 Spool System Position Control

Once failure has occurred, we need to lift the robot clear of danger. To do this we need a position controller. This controller is difficult because when controlling the position of the robot we have to account for the non linearity of the system. Specifically the controller should avoid entering regions where the robot lifts off the spring. To do this we will take the desired position and generate a smooth trajectory with a limited acceleration and a limited velocity. The acceleration limit should keep the robot in contact with the spring, while the velocity limit is mainly for the comfort of the robot operator. Once we have the smooth trajectory, a PD controller with feed forward will generate motor velocity commands.

The smooth trajectory is generated by a set point filter. The core idea is to use a critically damped second order system that responses to an input. This filter's transfer function is:

$$\frac{y_{out}}{y_{in}} = \frac{1}{(s/\tau)^2 + 2(s/\tau) + 1} = \frac{1}{(s/\tau + 1)^2} = \frac{k}{s^2 + bs + k}$$

Where $k = \tau^2$ and $b = 2\tau$. This works well at generating a smooth trajectory, but it does not limit the velocity or acceleration. This is added in the block diagram of the filter. The non limiting block diagram of the filter can be seen in Figure 6.

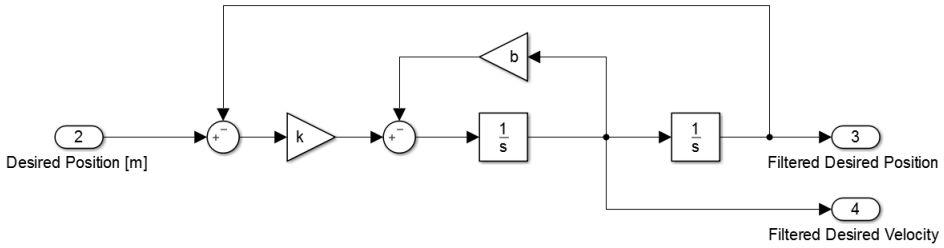

Figure 6: The caption of a single sentence does not have period at the end

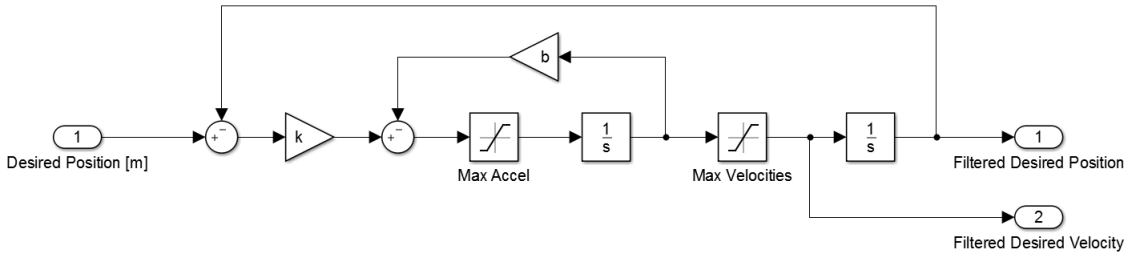

Figure 7: The caption of a single sentence does not have period at the end

Now, we can limit the maximum acceleration and maximum velocity by adding in saturation blocks to the block diagram. As their name suggests there are blocks that limit the signal passing through them to be less than a maximum value and more than a minimum value. The limiting block diagram can be seen in figure 7. To compare these two filters we can examine their reaction to a large step input in figure 8

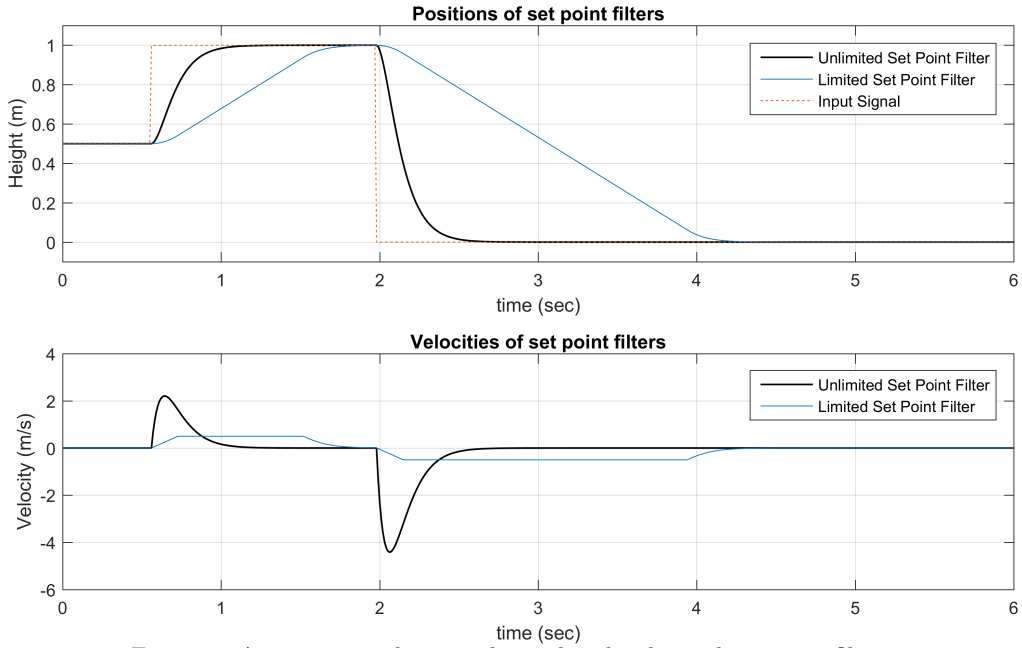

Figure 8: A comparison between limited and unlimited set point filters

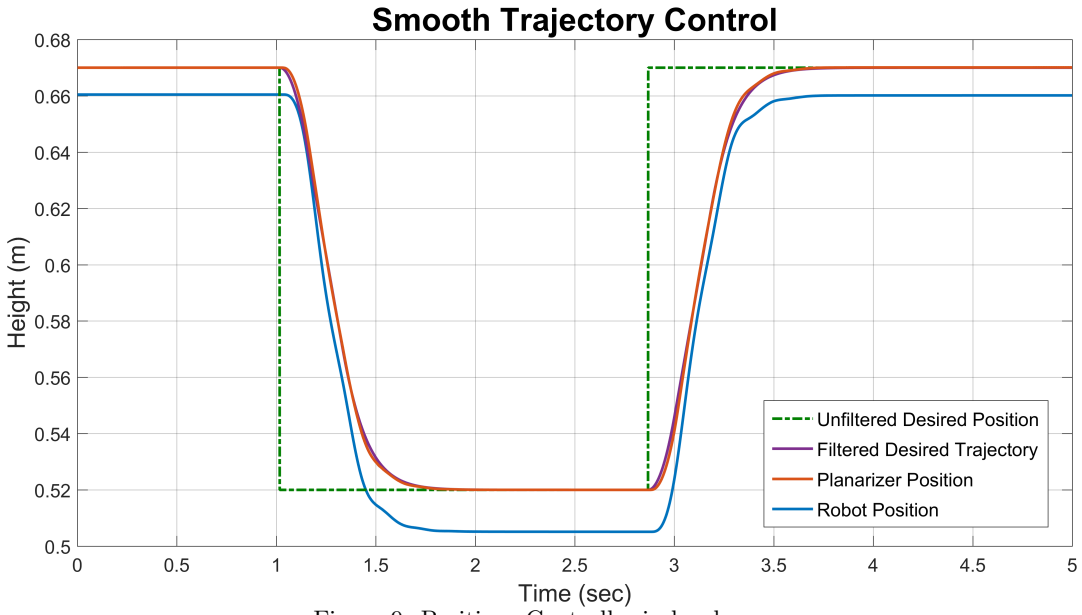

Figure 9: Positions Controller in hardware

A PD with feed forward controller then makes the planarizer system follow this desired trajectory. The equation of the controller is shown below.

$$\dot{y}_{motor} = k_p(y_{des} - y_p) + k_d(\dot{y}_{des} - \dot{y}_p) + k_{\text{ff}}\dot{y}_{des}$$

Looking at this full controller implemented in hardware we see very good behavior. This behavior is shown in figure 9. The motor tracks the desired trajectory very closely. There is no overshoot in planarizer position. The robot follows the planarizer position closely with minimal oscillations. This is exactly what we hoped to see.

4 Conclusions

After implementing these systems in hardware we found that they were quite effective. The fault sensing is robust and effective. The shadowing controller has a very minimal effect on the walking robot's dynamics. The position controller generates a smooth trajectory that easily and safely lifts the robot out of harms way before it fails. This system should enable more effective use of on line learning experiments.

5 Future Work

This system has the capability to do more than just failure recovery. It should be able to apply a commanded vertical force to the robot. By controlling the spring's deflection, the force should be reasonably straightforward to regulate. Once a force controller is implemented, it opens the door to interesting new experiments. By applying a constant upward force, the plagiarizer should be able to artificially lower the gravitational force on the robot. This could be very desirable if the robot encounters actuator torque limits during its gait. With lower gravity, the robot should be able to use less motor input to achieve a similar gait.

References

[1] Manuel F Silva and J.A Tenreiro Machado. A historical perspective of legged robots. *Journal of Vibration and Control*, 13(9-10):1447–1487, 2007.

[2] C. David Remy. *Optimal exploitation of natural dynamics in legged locomotion.* PhD thesis, ETH Zurich, July 2011.

[3] Brian A. Kyckelhahn and Brian A. Kyckelhahn. Jitter in self-explanatory simulation. *Proceedings of the 18th International Qualitative Reasoning Workshop*, 2004.

Appendix A: Model Derivations

We want to identify what conditions are required to maintain $y_r \geq y_p$. To simplify the analysis we will separate the elastic and damping force in the spring.

$$F_{sp} = F_k + F_b$$

Where

$$F_b = \begin{cases} b(\dot{y}_p - \dot{y}_r) & y_p > y_r + \epsilon \\ b(-2(\frac{y_p - y_r}{\epsilon})^3 + 3(\frac{y_p - y_r}{\epsilon})^2)(\dot{y}_p - \dot{y}_r) & y_r \leq y_p \leq y_r + \epsilon \\ 0 & y_p < y_r \end{cases}$$

We will treat the damping as a disturbance that we can identify a bound for. If we let $B \geq max(|b(\dot{y}_p - \dot{y}_r)|)$, we can say that

$$F_b \leq B$$

The result of this is that we can conclude that for all y_r and all y_p that

$$F_{sp} \geq F_k + B$$

We want to find the conditions where $y_r \geq y_p$ or alternatively where $f_k > 0$. Looking back to the equation of motion,

$$M_r \ddot{y}_r = F_k + F_b - F_g$$

$$M_r \ddot{y}_r + F_g - B = F_k$$

Applying the condition that $f_k > 0$

$$M_r \ddot{y}_r + F_g - B > 0$$

$$\ddot{y}_r > -g + \frac{B}{M_r}$$

We get our final expression for what the robot acceleration must be to keep the robot from lifting off of the planarizer.

Next, we want to get an actual estimate for what B is. We know that $M_r = 10.5kg$ and that the spring constant $k = 5254N/m$. We observe heavily damped oscillation in the robot when it is dropped onto the spring. From this we know that the damping ratio is less than one, so we conservatively say that it is equal to one.

$$\zeta = 1 = \frac{b}{2\sqrt{kM_r}}$$

$$2\sqrt{kM_r} = b$$

$$b = 2\sqrt{5254 * 10.5} = 470\frac{N}{m/s}$$

When controlling the velocity of the robot we will trying to move the robot and the planarizer together. Becuase of this we can assume their velocities are relatively close together, differing by at most 0.1 m/s. This means our damping force is at most $47N$. Finally we find that our requirement for linear operation is:

$$\ddot{y}_r > -5.11\frac{m}{sec^2}$$