Rachel Cleaveland
Professor Andre Platzer
Logical Foundations of Cyber-Physical Systems
December 6, 2019

Modelling Safe and Efficient Tumbles of An Acrobatically Inclined Robot

Abstract

This system described in this paper models the motion and control decisions of a robot as it performs a series of flips and twists through the air, like a gymnast. Specifically, when a gymnast is performing a tumbling pass, she first bounces off of the ground with some initial upward velocity, and then flips and twists until she hits the ground. This presents an interesting safety consideration because not all tumbles are the same: many factors could result in a gymnast getting a poor bounce off the ground, and differences in each gymnast's technique and physique could lead to different rotational speed in the flips and twists as she performs her tumble. The goal of this system is to abstract away the uncontrollable variables when doing tumbles to see if no matter what the circumstances of its initial bounce and its ability to generate rotation, a robotic gymnast will always be able to land with a proper orientation to the ground. The model included a controller to allow the robot to adjust the speed of its flips and twists, and KeYmaera X was used to reason about the safety of the system. The model was ultimately proven to be safe, meaning that no matter the initial bounce that the robot gets off the ground, it will be able to return to its feet safely.

I. Introduction

October 13, 2019 marks the day that Simone Biles broke, and then rebroke, the world record for the most medals won by a single gymnast at the world championships. With 25 medals, Simone Biles has proven that she is the greatest gymnast of all time. Her strength and impeccable air-awareness allows her not only to perform the most difficult skills possible in the sport of gymnastics, but also to land on her feet every time. This then begs the question: how is Simone able to land on her feet despite the inevitable inconsistency in her bounce coming off of the floor? How is she able to do skills with a variable number of flips and twists and still land safely?

This scenario can be transformed into a cyber-physical system as follows. Consider a robot that is tumbling backwards. Given the bounce that it gets off of the floor going into its tumbling pass, it must determine how many flips and twists are safe to complete while ensuring a feet-first landing. The robot should also land facing either forward or backward: no quarter twists should be allowed, as landing in this position presents a great risk of damaging the robot's knee joints.

This system is interesting to consider in the context of this class because in previous labs, we have been able to control the speed of a car driving in circles. In this system, on the other hand, once rotation has started it is physically unrealistic to be able to slow one's linear speed when flipping through the air. What gymnasts do instead is that they adjust the distance of their limbs to their center of gravity in order to change their moment of inertia, thereby altering their rotational velocity without adjusting their linear velocity. In addition, once a gymnast starts rotating, she cannot realistically stop her rotation because she would have to make her moment of inertia infinite. The decision whether or not to flip, and how fast to rotate, becomes trickier without this fallback option to stop rotation.

Another aspect of this system that makes it challenging to study is its use of circular motion from the flipping and twisting. A typical controller for a cyber-physical systems involving robots would make its control decisions by assessing the current location of the robot, as well as the future location or trajectory of the robot if the controller were to make a decision. The controller can then decide if this future location or trajectory is safe, which will lead the controller to ultimately make a control decision. However, whenever circular motion is present in a model, this complicates the controller because it is much more difficult to adequately predict the future location of the robot. KeYmaera X currently does not support trigonometric functions, so this model is challenging to consider due to the fact that the controller cannot simply precompute the exact position of the gymnast at a given time.

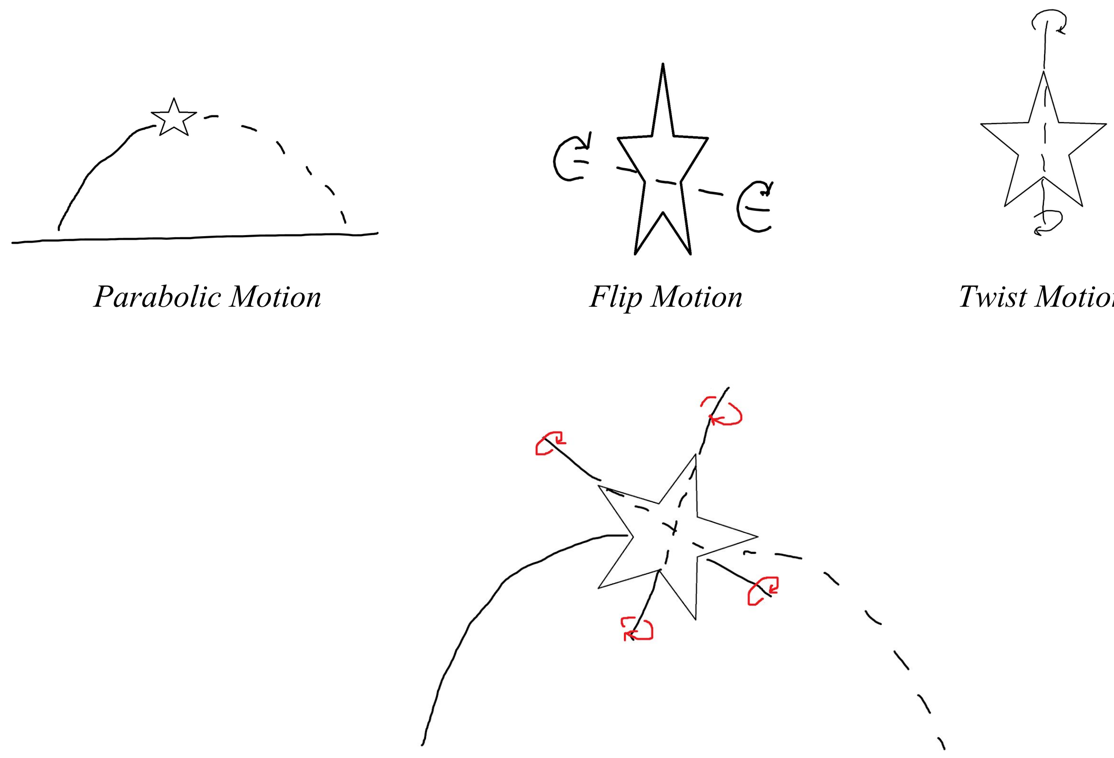
II. Related Work

This system is also interesting to consider from a broader perspective because it is relevant in the field of self-righting robots. Prior research done by Shields et al. [2013] shows a model of a cat robot that is able to right itself during a fall. The biggest difference between my project and this sort of model is that their model was verified using simulation as opposed to with a theorem prover like KeYmaera X, and their model also does not consider the amount of time the cat would take to flip over. My project therefore slots in nicely to Shields et al.'s work because their research would ensure that the cat is able to flip itself over, and my project would ensure that it times its flip properly so that it lands on its feet and does not overshoot or undershoot a safe landing position.

This project also complements other research in the field of self-righting robots because while many papers in addition to Shields et al. describe self-righting mechanisms of specific animals and robots, this project abstracts away from the specific type of robot being considered and is therefore applicable to those scenarios. Ribak and Weihs [2011], for instance, look at the self-righting mechanism of Click Beetles, which launch themselves into the air without the use of their legs, thus controlling the velocity with which they hop off of the ground, but not the angle. This means that Click Beetles can launch themselves into the air but are not capable of controlling their orientation upon landing. Saranli and Koditschek [2002] explore modelling and simulating a backflip of a hexapedal robot with a controller that considers the forces acting on the feet of the robot before takeoff. In both of these cases, great care is taken to model the physics of the specific robots being considered, but the findings are as a result highly specific to those robots. Since my project abstracts away the details of the specific robot being considered, its result can be more broadly applied to the field of self-righting robots.

III. The Motion: Assumptions and Simplifications

There are three aspects to consider in the motion of an acrobatic robot as it performs a tumbling pass: the parabolic motion of the robot as it travels up from and back down to the ground; the rotational motion about its horizontal axis as it performs a flip; and the rotational motion about its vertical axis as it performs a twist. These elements of motion are diagrammed in Figure 3.1.



Parabolic Motion *Flip Motion* *Twist Motion*

Combined Motion

Figure 3.1.

The first major simplification of this model is the separation of these three pieces of motion. This is a realistic simplification because a gymnast generates these pieces of motion independently. She generates the parabolic motion by bouncing off the ground with some initial upward velocity; she generates the backwards rotational flipping motion by tucking her legs towards her chest once she leaves the ground; and she generates the rotational twisting motion by tucking her arms towards her body and looking over her right shoulder after leaving the ground. The generation of the parabolic motion happens first, and the rotations are generated afterwards, with the flip being generated by the lower half of her body and the twist being generated by the upper half of her body.

Additionally, the ways that a gymnast controls these elements of her motion do not overlap. Once she leaves the ground for her parabolic motion, she is at the mercy of gravity and therefore has no way of altering the amount of time she has until returning to the ground. Once she starts her flipping, she can control how fast she flips by bringing her legs closer to, or farther from, her body, which changes her moment of inertia and therefore affects the number of flips that she can do in the same span of time, as exemplified in the equation in Equation 3.1.

$$E_{rotational} = 1/2\, I\omega^2$$

Equation 3.1. The amount of rotational kinetic energy in the system, where I is the moment of inertia and ω is the angular velocity.

The amount of rotational kinetic energy does not change because there are no significant forces acting on a gymnast as she is flipping, so a decrease in her moment of inertia will therefore increase her angular velocity, and an increase in her moment of inertia will decrease her angular velocity. This applies to both the flipping motion and the twisting motion of the robot as it is completing its tumble: while the robot cannot change the rotational kinetic energy of its flipping or twisting motion, it can change its angular velocity around both of its rotating axes by adjusting the position of its limbs with respect to its axis of rotation.

Additionally, it can control its moment of inertia with respect to its flipping axis independently of its moment of inertia with respect to its twisting axis. Consider a robot tucking its legs to flip faster. While it now has more of its mass closer to the axis of its flip, which cuts somewhere across its hip region, it does not alter the distribution of its mass relative to its twisting axis, which cuts straight down the length of its body. Thus, the angular velocity of its flipping motion can be altered without affecting the angular velocity of its twisting motion, and vice versa. Therefore, since each aspect of motion is generated and controlled independently by the gymnast, it is realistic for these pieces of motion to be modelled independently as well.

The second major simplification is in the modelling of the rotational motion of the flips and twists. Fundamentally, a flip or a twist is a spin in a circle, so the model of the flips and twists should be built upon circular motion. Ultimately, many aspects of the robot that affect how it flips can be abstracted away; they add complexity to the proof without adding insight. What is important to consider is that the robot has a certain radius from its head to its center of gravity when it enters a tuck position to flip, and it has a certain radius from its shoulder to the central axis of its body as it performs its twist. Additionally, its head and shoulders have their own linear velocities once they start the tumble. Thus, the flipping and twisting motion can be simplified to modelling the motion of a point moving clockwise around a circle. The parametric equations of this motion are presented in Equation 3.2.

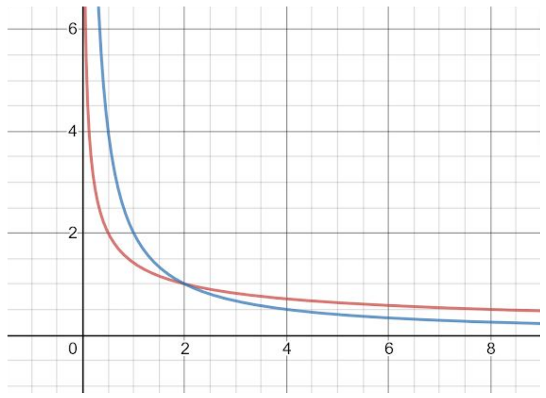
$$x = r\,sin(v/r * t)$$
$$y = r\,cos(v/r * t)$$

Equation 3.2. Parametric equations of a circle.

These equations start with $x = sin(t)$ and $y = cos(t)$, to capture the clockwise motion. They are multiplied by the radius r to alter the amplitude of the trigonometric functions, and the time variable is multiplied by v/r to alter the period of the circular motion.

The final simplification is in the modelling of the gymnast's control of her rotation. As previously described, a gymnast controls the speed of her rotation by bringing in or splaying her limbs, thereby changing her moment of inertia, which is inversely correlated to her angular velocity. It is not necessary in this model, however, specify the particular inverse correlation between its moment of inertia and angular velocity. This is because it can be assumed that the robot is capable of bringing in or splaying its limbs very quickly, just as a real gymnast is capable of, so the continuous change in its moment of inertia does not need to be considered, only the starting point and ending point. Therefore, any function that reflects an inverse correlation between the moment of inertia of the robot and its angular velocity suffices.

Thus, in this model, the radius of the circles used to model the flipping and twisting motion are used and modified to reflect the changing moment of inertia of the robot. As can be seen in Figure 3.2, angular velocity is inversely correlated to moment of inertia as well as radius, and the functions follow a very similar shape. Therefore, it is valid to consider changing the radius of the circular motion of the flip and twist to adequately model the effect of an acrobatic robot bringing in or splaying its limbs to control its moment of inertia, and thus its speed of rotation.



In red: $y = \sqrt{\frac{2}{x}}$

In blue: $y = \frac{1}{x}$

Figure 3.2. Angular velocity as a function of moment of inertia shown in red, angular velocity as a function of radius shown in blue.

IV. The Model

The assumptions and simplifications of the model given in the previous section interact as follows.

4.1 Preconditions

The preconditions encode the following assumptions.

- The robot bounces off of the ground and has some positive amount of time until it returns to the ground, given in the model as the variable *timeToGround*. The use of this variable abstracts away the parabolic motion of the tumble, so this aspect of motion will not be included in the differential equations in Section 4.3.
- The robot has some positive range of radii that the motion of its flips and twists can take, given in the model as *minflipr, maxflipr, mintwistr,* and *maxtwistr*. This makes the model more realistic, as a gymnast will not have unbounded control of her moment of inertia due to the unchangeability of the length of her limbs.
- The variables pull and wrap are positive constants that represent the linear velocity of the flip and twist, respectively, if the gymnast choosing to start flipping and twisting.
- The robot starts facing forwards and in an upright position, as expressed by the conditions $flipy = flipr$, $flipx = 0$, $twisty = twistr$, and $twistx = 0$.

4.2 Controller

There are two aspects to the controller: the control decision before the loop and the control decision during the loop. The most important aspect of this model is that once the robot begins flipping or twisting, it has no way to halt its rotation, which is reflected in the model by the bounds placed on the radii that the robot can choose. Thus, it is imperative to the safety of the model that if the robot cannot land safely after *timeToGround* time for any possible flip or twist radius, it should not start flipping or twisting in the first place. The robot cannot decide to do a single flip and then halt its rotation if it thinks that doing a second flip would be unsafe, but the controller must still allow a no-flip or no-twist option to the robot. Thus, the first aspect of the controller is placed before the loop and thus before any motion occurs, and it gives the robot the option not to flip or twist if doing so would be unsafe.

The condition to determine if starting to flip or twist would be unsafe involves computing the number of flips the robot can do if it uses *minflipr*, as well as if it uses *maxflipr* (and the same is done for the number of twists). Then, if the gymnast can do at least one additional flip with *minflipr* as with *maxflipr*, it must be the case that the robot can end at any point on the circle by choosing a specific radius in the range [*minflipr*, *maxflipr*]. Thus, there must be some flip radius in that range that will put the robot perfectly upright after exactly *timeToGround* time, so it is safe for the robot to start flipping. The same argument can be made for the twisting case, the only difference being that there are two points on the circle that the robot can safely end at, which are exactly half a rotation apart, so it need only be the case that *mintwistr* results in doing a full additional half twist compared to *maxtwistr*. If this safety condition is not met, then the controller will tell the robot to set the linear velocity of its flipping or twisting motion to 0, and the robot will not begin rotating.

Note that computing the number of rotations that can be done in *timeToGround* time requires calculating the period of the circle, and calculating the period of a circle requires using π. However, π is irrational and thus cannot be precisely represented in KeYmaera X. Instead, I use two approximations of π, one overapproximation (3.14160), and one underapproximation (3.14159). The overapproximation is used to calculate the number of flips done using *minflipr*, and the underapproximation is used to calculate the number of flips done using *maxflipr*. Since these calculations involve dividing by π, the number of flips done using *minflipr* was underapproximated while the number of flips done using *maxflipr* was overapproximated. Since we require the number of *minflipr* flips to be at least one greater than the number of *maxflipr* flips, these opposite approximations do not jeopardize the safety of the system because they ultimately only increase the cases where the robot chooses not to start flipping, meaning they treat some safe cases as being unsafe, but do not treat any unsafe cases as safe.

The other aspect of the controller occurs in the loop body, once the gymnast has already started her flight through the air. This controller is event-triggered, and it wakes up every time the robot returns to an upright position from flipping, or when the gymnast returns to a position facing forwards or backwards while performing her twist. Before the differential equations, the controller nondeterministically assigns a value to flipr and then checks that this value is in the range [*minflipr*, *maxflipr*]. If this is true, it then checks that the selected value is safe by using a Taylor series approximation for the value of flipy after *timeToGround* time. Since the Taylor series for cosine involves summing terms of opposite signs, as shown in Equation 4.1, a lower bound can be achieved by cutting off the sum after a term is subtracted. This led to the Taylor series approximation for flipy, shown in Equation 4.1, used in the controller. The multiplication and division by the factors of *flipv* and *flipr* reflect the alteration of the period in this system, with the starting point being *flipy* = *flipr* and *flipx* = 0. The implications of this approximation will be discussed in further detail later.

$$\begin{aligned}\cos x &= 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} - \dots \\ &= \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n}}{(2n)!}\end{aligned}$$

Equation 4.1. Taylor series approximation for cosine.

The twisting case is slightly more involved because it requires an upper bound as well for *twisty*, since both *twistr* and *-twistr* are safe landing positions. This, however, can be achieved by negating the Taylor series approximation for the upper bound.

Once these upper and lower bounds are determined, plugging *timeToGround* into these approximations gives bounds for the values of *flipy* and *twisty* when the robot reaches the ground. Then, we know that the particular value of *flipy* or *twisty* chosen is safe if the bounded approximations are within some safe range of the proper landing positions. It was decided that a safe landing position would be within a 45 degree angle of upright or facing forwards or backwards, which means that *flipy* should be atleast *0.7 * flipr*, which is approximately $\sqrt{2}$ *flipr*, and *twisty* should be at least *0.7 * twistr*, or at most *-0.7*twistr*. Thus, because an approximation is used to determine the final values of *flipy* and *twisty*, it is no longer reasonable to require the gymnast to land in a perfectly upright position facing precisely forwards or backwards. Rather, it suffices to ensure that the robot would land nearly upright and nearly forwards or backwards. This is a realistic safety property as well because in reality, a gymnast can tolerate landing in a range of positions without risking her safety.

4.3 Differential Equations

As explained in Section 3, the motion of the flipping and twisting can be modelled using parametric equations of circular motion. Then, we can take the derivative of these parametric equations from Equation 4.1 to produce the ordinary differential to be used in the model, shown in Equation 4.2.

$x' = v\, cos(v/r\ * t) = (v/r)\ y$
$y' = -v\, sin(v/r\ * t\,) = (v/r)\ x$
Equation 4.2.

The evolution domain constraint requires that the *timeToGround* be at least 0, so that the robot does not continue flipping once it has reached the ground, and it also requires that *t* is either 0, or *twistx* not equal 0, or *flipy* not equal *flipr*. These conditions allow the controller to be event-triggered, so that the ODE stops evolving if the robot reaches an upright position in its flip or is facing forwards or backwards in her twist, but only once the system of ODEs has already started evolving and the gymnast has completed at least one full additional flip, or one additional half-twist.

4.4 Invariants and Postconditions

The loop invariant used is simple, and it is the same as the postcondition. That is, if *timeToGround* is 0, meaning that the robot is on the ground, then its position should be within approximately a 45 degree angle of being perfectly upright, and it should be within a 45 degree angle of facing perfectly forwards or backwards. These two aspects are the hallmarks of a safe landing, so it is important that they be true precisely when the robot hits the ground. However, during the tumble, when *timeToGround* is not 0, there is no simple invariant to relate the *timeToGround* to the gymnast's position in the air, because she could have a range of positions that are safe due to her ability to change her flipping and twisting radius, and because determining if her position is safe requires pre-computing her future position given her possible range of flipping and twisting radii, which is not efficient.

In opting for this simple loop invariant, we do not have as strong preconditions for proving each individual iteration of the loop to be safe, but we get around this by introducing more vacuous behavior in the controller. The flipping case will be detailed, but assume a parallel approach is taken in the twisting case. In the controller, we nondeterministically assign a value to *flipr*, and then discard runs in which the flip radius is not within the allowable range [*minflipr*, *maxflipr*], as well as flip radii in which will not put the robot back to a favorable landing position in *timeToGround* time (the method for approximating the robot's landing position after timeToGround time will be discussed later). Additionally, we also discard runs in which in the beginning of the loop, the robot is not perfectly upright, which facilitated the proof greatly for reasons that will be discussed shortly. Clearly, most runs of the system will be discarded due to the strong requirements on the flip radius as well as the robot's position going into each loop. However, the diamond modality proof sketch of liveness, which will be discussed later, sketches a proof for showing that there exists at least one run that will allow the robot to complete some number of flips and twists if it initially safe to do so, so it is realistic to accept such vacuous behavior in the controller because it does not destroy the liveness of the system.

V. Proof

Proving the safety of this model requires the use of a system with a box modality, and proving the liveness requires the use of a system with a diamond modality.

The proof of the box modality starts with the unfolding of our assumptions, which breaks the proof into four parts: the case where we choose to flip and twist, the case where we choose to flip and not twist, the case where we choose to twist and not flip, and the case where we choose neither to flip nor twist.

The next step is to apply the loop invariant, and we have now converted the original proof to a loop induction proof in each of the four cases. The base case and exit case are trivial to prove: *timeToGround* is initially non-zero, so the base case closes easily, and the loop invariant is identical to the postcondition, so the exit case closes as well. Additionally, the non-flipping, non-twisting case is trivial to prove because the robot leaves the ground in a favorable landing position, and its position does not change because it chose not to flip or twist. Thus, its landing position is identical to its starting position, and the robot always lands safely.

The other cases, however, required a much more intricate approach. KeYmaera X, the theorem prover used to prove this model, does not support trigonometric functions, which are necessary to model the exact positions of *flipx* and *flipy* at particular times when modelling circular motion. This complicated the controller because the controller needed a way to precompute the final position of *flipy* after *timeToGround* time in order to determine if the nondeterministically-chosen value for flipr was safe to use. As previously discussed, the controller circumvented this problem by using a Taylor series approximation to apply a minimum bound to *flipy* and *twisty*.

The proofs of these cases were completed as follows: we were able to assume from the controller that our flipr would allow the Taylor series approximation of flipy to be at least *0.7*flipr* after *timeToGround* time. Then, in the ordinary differential equation, we were able to do a differential cut in that *flipy* was always at least the Taylor series approximation. This closed this branch of the proof easily because we have in the evolution domain constraint that *flipy* is at least the Taylor series approximation for all values of time, and we have as a precondition that the Taylor series approximation is at least *0.7*flipr* after *timeToGround* time, thus the prover can conclude that when *timeToGround - t = 0*, meaning when *timeToGround = t*, the Taylor series approximation must be at least *0.7*flipr*, and thus *flipy* must be at least *0.7*flipr*.

The other branch of the proof created from doing this differential cut is the proof that after all runs of the hybrid program, it is true that the Taylor series approximation provides a lower bound for *flipy*. The proof of this was simple, yet extremely tedious. It involves an extended series of differential cuts of the derivative of the Taylor series, followed by differential invariants to close each proof branch. After repeating this process many times, the final branch simplified to $flipy \leq flipr$, which was trivial to prove.

As for the liveness of the system, which is represented in the diamond modality proof, a proof sketch can be found in the included PDF. To summarize, loop convergence should be used to prove the post condition. The loop convergence property, $p(\tau)$, should essentially state that *flipy* is equal to *flipr*, and τ is the number of flips remaining until reaching the ground and is a whole number. Then, by the preconditions, we know that *flipy* is initially *flipr*, and there must be some *flipr* such that τ exists. If $p(\tau)$ for some $\tau \leq 0$ is eventually true, then *flipy = flipr* and there are 0 flips remaining until reaching the ground, meaning that timeToGround is 0 and we are in a safe landing position.

This is presented as a proof sketch and not a formal proof for several reasons, discussed in Section 6.

VI. Conclusion

The main project goal, which was the proof of the model using the box modality found in SimoneBilesFinal.kyx, was met. The proof closed, and the model did not lose too much expressivity in the process. Both flipping and twisting were proven to be safe, and the model incorporated both options to flip and not to flip, depending on the *timeToGround* value, which were my biggest goals. Additionally, a proof sketch was achieved to prove the liveness of the controller for the flipping and twisting case. This is a partial success, as the proof sketch is intuitively clear and could likely be mathematically rigorized with a couple of small tweaks to the model, which are discussed in the TermProjectDiamond.pdf file. However, given the proof sketch's use of the floor function, it cannot be formally proven in KeYmaera X, as will be discussed further in the paper.

The largest impediment to completing the box proof, as expected, was the fact that there was no safe default case on which to fall back on in the controller. This is a characteristic of tumbling, as once a gymnast starts flipping and twisting, there is almost nothing she can do to change the linear speed of her rotation, so a gymnast cannot simply decide to stop flipping or twisting whenever she sees fit. This is clear when observing gymnastics sporting events, as multiple gymnasts will land tumbling passes poorly or dangerously, even at an elite level of gymnastics. Thus, designing a controller that could guarantee safety while not sacrificing the liveness of the system was a major challenge. Particularly, it was difficult to design a controller that could adequately determine the existence of a safe and valid value for *flipr* and *twistr*, as well as adequately compute the values of *flipy* and *twisty* that would result from using these values of *flipr* and *twistr* for *timeToGround* time.

Preliminary ideas of the controller relied on using the *maxflipr* and *minflipr* range to determine safety. Specifically, if the range of landing positions that the robot could achieve from all flipr in the range [*maxflipr*, *minflipr*], spans the upright position, then the controller will have a choice of a flip radius that it knows will put the robot in a safe landing position. After pursuing this approach, however, it became clear that expressing the idea of two values spanning a natural number was very difficult to represent in KeYmaera X. I had wanted to make use of the floor function to express this notion, by requiring that the floor of the number of flips that the robot can do by spinning as fast as possible as strictly greater than the number of flips that it can do spinning as slowly as possible. I learned, however, that the floor function is not supported in KeYmaera X for the following reason.

By Godel's first incompleteness theorem, first-order logic (FOL) over the natural numbers is undecidable. In fact, it is not even semidecidable, and as a result, functions like the floor function, which implicitly incorporate natural numbers into the proof, should not be used in theorem provers like KeYmaera X. To get around this obstacle, the safety condition to start flipping or twisting was made to be very strong: if the robot cannot do a full additional flip with *minflipr* compared to *maxflipr*, then it should not start flipping at all. It need only be the case that some flip radius in the range [*minflipr*, *maxflipr*] return the robot to an upright position, but this is not easily expressible in KeYmaera X without introducing FOL over the natural numbers, so the stronger safety condition had to be used.

Also, the Taylor series approximation was used to approximate final values for *flipy* and *twisty* in the absence of being able to use trigonometric functions in KeYmaera X. It should be noted that the proof is it stands contains a major source of error, in that the Taylor series approximation used in the controller only tightly approximates the cosine function to about half of a rotation, shown in Figure 6.1.

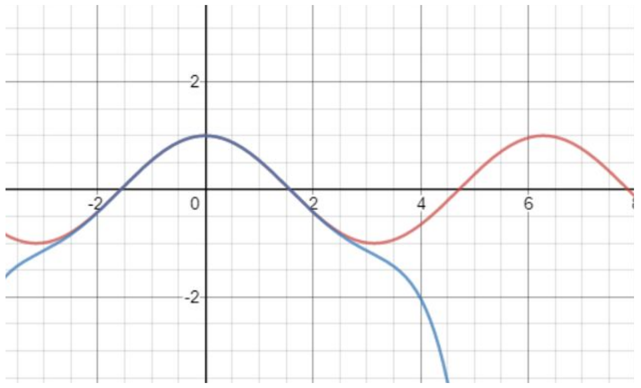


Figure 6.1. $1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!}$ shown in blue, cos(x) shown in red.

The controller therefore has strong vacuous behavior past about half a rotation because the approximation of *flipy* becomes very negative and therefore cannot be greater than or equal to *0.7*flipr*, for any positive *flipr*. However, this decision was made intentionally to reduce the tedium in completing the proof. Every differential cut in of the derivative of the Taylor series introduced more and more terms, as shown in Figure 6.2, so even completing the proof for the approximation out to t^6 took a very long time. However, adding more terms to the Taylor series approximation would absolutely still result in a proof of the model.

dC 1: establish the Taylor series lower bound on flipy

$$r - \frac{r(\frac{v}{r}t)^2}{2} + \frac{r(\frac{v}{r}t)^4}{24} + \frac{r(\frac{v}{r}t)^6}{720} \le \frac{v}{r} flipy$$

dC 2: cut in the derivate of the previous Taylor series, use be used by dI

$$-r\frac{v}{r}\left(\frac{v}{r}t\right) + \frac{r\frac{v}{r}(\frac{v}{r}t)^3}{6} + \frac{r\frac{v}{r}(\frac{v}{r}t)^5}{120} \le -\frac{v}{r}\frac{v}{r} flipx$$

dC 3: cut in the derivate of the previous series, use be used by dI

$$-r\frac{v}{r}\frac{v}{r} + \frac{r\frac{v}{r}\frac{v}{r}(\frac{v}{r}t)^2}{2} + \frac{r\frac{v}{r}\frac{v}{r}(\frac{v}{r}t)^4}{24} \le -\frac{v}{r}\frac{v}{r}\frac{v}{r} flipy$$

Figure 6.2. Three out of the seven differential cuts used to prove the induction step of the loop invariant proof.

Specifically, in the case of a female gymnast, it is realistic to assume that she will not be able to complete more than two flips and four twists in a single tumbling pass, so we need only use enough terms in the Taylor series approximation. However, to tightly approximate a full two periods of a circle, the Taylor series approximation would have to go up to the term t^{34} , as shown in Figure 6.3. This would clearly result in a much more accurate controller, but for my purposes and for the purposes of this class, the executive decision was made not to spend the time completing this proof, because the control flow of proving the Taylor series approximate was already demonstrated with the proof out to t^6.

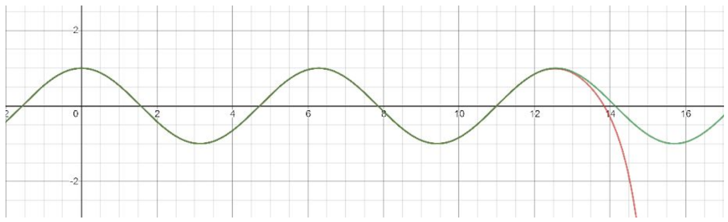


Figure 6.3. cos(x) shown in green, and the Taylor series approximation out to t^{34} shown in red.

It should be noted that in the case of a robotic gymnast, it is reasonable to use a Taylor series approximation because there is a realistic upper bound on the number of flips and twists that a gymnast can perform, and thus the Taylor series approximation does not have to be accurate beyond a few periods of the cosine function. However, this proof technique would no longer be useful if there were no assumptions about the maximum number of flips and twists that a robotic gymnast could perform, as the Taylor series will always diverge from the cosine function and lead to vacuous behavior in the controller.

Acknowledgements

I would like to thank Dr. Andre Platzer, whose insights on the Taylor series approximations as well as the undecidability of first order logic of the natural numbers gave me great direction throughout this project. I would also like to thank Katherine Cordwell, also helped to direct this project in a better path. Also, major thanks to Yong Kiam Tan, whose knowledge of proof techniques and KeYmaera X allowed me to close the proof, and to Stefan Mitsch for his additional knowledge of KeYmaera X.

Works Cited

Ribak, G., & Weihs, D. (2011). Jumping without Using Legs: The Jump of the Click-Beetles (Elateridae) Is Morphologically Constrained. *PLoS ONE*, *6*(6). doi: 10.1371/journal.pone.0020871

Saranli, U. and Koditschek, D. (2002). Back Flips with a Hexapedal Robot, presented at Proceedings of the 2002 IEEE International Conference on Robotics & Automation, Washington, DC, 2002. Ann Arbor: University of Michigan.

Shields, B., et al. (2013). Falling cat robot lands on its feet, presented at Australasian Conference on Robotics and Automation, Adelaide, 2013. Sydney, Australia: University of New South Wales.

Diamond Modality Proof Sketch

Refer to the Diamond .kyx file for the model referred to by $A \vdash \langle\alpha\rangle Q$

$$\frac{A \vdash \exists \tau p(\tau), \Delta \qquad \vdash \forall \tau > 0 (p(\tau) \rightarrow \langle\alpha\rangle p(\tau - 1)) \qquad \exists \tau \leq 0 p(\tau) \vdash Q}{A \wedge minflipr \leq flipr \leq maxflipr \vdash \langle\alpha\rangle Q} \text{ (con)}$$

Only the proof sketch of the flipping aspect of the model is shown, since the twisting aspect proof sketch would be nearly identical.

Let $p(\tau) =$

$$\exists flipr \quad minflipr \leq flipr \leq maxflipr \wedge \text{flipy} = flipr \wedge \tau = \frac{\text{timeToGround} * \text{flipv}}{2\pi flipr} \wedge \text{floor}(\tau) = \tau$$

Intuitively, $p(\tau)$ is saying that there exists some valid value for the radius flipr such that flipy is initially equal to flipr (meaning the robot is upright), τ is equal to the number of flips that the robot can do in timeToGround time given the value for flipr, and τ is a whole number.

$A \vdash \exists \tau p(\tau), \Delta$
This is clearly true initially because one of our preconditions is that flipy is equal to flipr, and another is that the robot can do at least one additional flip if it uses its minflipr compared to its maxflipr when it flips for timeToGround time. If this is the case, then there must be some intermediate value for flipr that puts the robot perfectly upright in exactly timeToGround time. Note the slight modification from the SimoneBilesDiamond.kyx file, which is the addition of the precondition that minflipr $\leq$ flipr $\leq$ maxflipr. This should have been added to the preconditions in SimoneBilesDiamond.kyx in order to prove this initial $p(\tau)$ condition.

$\vdash \forall \tau > 0 (p(\tau) \rightarrow \langle\alpha\rangle p(\tau - 1))$
We know initially from $p(\tau)$ that flipy is equal to flipr, so the robot is upright at the start of the run of the hybrid program α. And, we know that this particular value of flipr ensures that the robot does a whole number of flips before returning to the ground, in timeToGround time. Then, all that must be the case is that the ODE is able to run for exactly the amount of time equal to the period of evolving around the circle. This is because running the ODE for the period of the circle will put the robot exactly upright once again, since it started in the upright position. Then, the robot will have completed one additional flip, meaning that it will have one fewer flip before reaching the ground, and $p(\tau - 1)$ must be true. Note an additional modification that should have been made to the model in SimoneBilesDiamond.kyx: the evolution domain constraint involving the position of flipy and twistx should be removed. If only the flip or only the twist were considered, the evolution domain constraints would not be a problem, but by combining the evolution domain constraints of the flip and twist, we can no longer ensure that any run of the ODE will run for at least a full period of the flip and twist.

$\exists \tau \leq 0 \quad p(\tau) \vdash Q$
If there exists a $\tau \leq 0$ such that $p(\tau)$, then intuitively the robot must be on the ground, since there are 0 periods that can happen in timeToGround time. Then, we know that flipy is flipr, so the postcondition that timeToGround$= 0 \rightarrow flipy \geq 0.7 * flipr$ is satisfied.