# Optimization-free Ground Contact Force Constraint Satisfaction in Quadrupedal Locomotion

Arjun Viswanathan
*MS Robotics: EECE*
*Northeastern University*
Boston, USA
viswanathan.ar@northeastern.edu

Rida Jawed
*BS Computer Engineering and Computer Science*
*Northeastern University*
Boston, USA
jawed.r@northeastern.edu

Rahul Sha
*MS Robotics: EECE*
*Northeastern University*
Boston, USA
pathepurshankar.r@northeastern.edu

Dilip Gummadidala
*MS Robotics: EECE*
*Northeastern University*
Boston, USA
gummadidala.d@northeastern.edu

***Abstract*—We will be studying the algorithm used in an optimization-free controller on a quadruped robot. This paper is published by Northeastern University and seeks control design paradigms that bypass expensive optimization based or learning based frameworks. They use a supervisory controller and an Explicit Reference Governor (ERG). The goal is to verify the algorithm works as claimed, and see if it makes sense to extend the base reduced order model to be more accurate to the actual robot, i.e adding more joints, leading to greater accuracy.**

**We picked this paper because it uses an optimization and learning free approach, which can both be very cost intensive in legged robotics. The amount of data and dynamics modelling required to ensure smooth motion is a complex process.**

***Index Terms*—Lyapunov stability, Ground Contact Forces, Explicit Reference Governor**

## I. Introduction and Motivation

Legged robot locomotion control typically consists of optimization based and learning based approaches. Unfortunately, both come with their own flaws as well as merits. Optimization based controllers adjust the gait parameters throughout the whole gait cycle, such that not only the robot posture is adjusted, but also the joint positions. But they rely heavily on initial conditions, and may not be adaptable to a variety of real world environments. On the other hand, learning based controllers are efficient and can learn a variety of environments, but require a lot of time and compute power to train and tune.

We propose to study this particular optimization-free approach to analyze the efficiency of the ERG. Many of today's real-time approaches have achieved better efficiencies through reduced-order models and decomposition methods, but they still rely on optimization. This project will introduce us to common pitfalls and challenges in the domain of legged robotics, while also presenting a new approach. We also learn about concepts learned in class like Lyapunov stability properties and linear algebra to map the coordinate transforms of each part of the legged robot.

## II. Problem Statement and Background

The literature provides a Reduced Order Model (ROM) using Euler-Lagrangian dynamics formulation to simplify the robot for the use case. This allows high fidelity testing of the locomotion accuracy while not having to worry about designing everything on the physical robot. We will highlight the original literature and code they present first, and then highlight our changes to the code to produce our results.

### A. Foot Positions

The ROM for Husky Carbon in the literature is a box for the base. There is no knee joint, but only the frontal and sagittal hip joints that have been modeled. The foot end positions are modeled through the following equation.

$$p_{fi} = p_B + R_B l^B_{hi} + R_B l^B_{fi} \tag{1}$$

Where $p_B$ is the position of the base in, $R_B$ is the rotation associated with the base, and $l^B_{fi}$ is the length to the foot end positions with respect to the base. The length of the foot end position from the base is determined by the following equation.

$$l^B_{fi} = R_y(\phi_i) R_x(\gamma_i)[0, 0, -r_i]^T \tag{2}$$

Where $\phi_i$ is the frontal hip angle, $\gamma_i$ is the sagittal angle, and $r_i$ is the prismatic joint length. The joint angles are in radians and the lengths are in meters.

### B. Lagrangian Equations of Motion

The kinetic energy of the system is modeled by

$$K = \frac{m_B \dot{p}_B^T \dot{p}_B + \omega_B^T \hat{I}_B \omega_B}{2} \tag{3}$$

which combines the linear and angular components. The potential energy of the system is then

$$V = -m_B p_B^T g \tag{4}$$

and so putting these together, we obtain the Lagrangian as $L = K - V$. This becomes relevant in calculating the equations of motion as shown below:

$$\frac{d}{dt}(\frac{\partial L}{\partial p_B}) - \frac{\partial L}{\partial p_B} = u_1 \tag{5}$$

which is a generalized vector that models the linear motion and

$$\frac{d}{dt}(\frac{\partial L}{\partial p_B}) - \frac{\partial L}{\partial p_B} = u_2 \tag{6}$$

which is a generalized vector that models the angular motion. With these vectors, the dynamic system acceleration can be written as

$$M[\ddot{p}_B, \dot{\omega}_B] + h = \sum_{i=1}^{4} B_{gi} u_{gi} \tag{7}$$

where $M$ is the mass inertia matrix, $h$ contains the Coriolis and gravitational vectors, and $B_{gi}u_{gi}$ represent the generalized force due to GRF acting on foot $i$. These components are also a function of leg joint variables $\phi_i, \gamma_i, r_i$ which are driven by setting their accelerations to track desired joint states. This is defined as

$$q_L = [\phi^T, \gamma^T, r^T]^T, \ddot{q}_L = u_L \tag{8}$$

$u_L$ forms the control input to the system in the force of leg joint state accelerations. So now combining the dynamic and massless leg states, we form the full system state as

$$x = [q_L^T \cdot \dot{q}_L^T, r_B^T, p_B^T, \dot{p}_B^T, \omega_B^T]^T \tag{9}$$

and now our equation of motion can be written as

$$\dot{x} = f(x, u_L, u_g) \tag{10}$$

where $x$ is the system state, $u_L$ is the control, $u_g$ is the GRF, and $dotx$ is the change in system state, and this is implemented in code.

## C. Ground Reference Forces (GRFs)

The GRF is modeled using a compliant ground model and Stribeck friction model as follows.

$$u_{gi} = \begin{cases} 0 & z_i > 0 \\ [u_{gi,x}, u_{gi,y}, u_{gi,z}^T] & else \end{cases}$$

where

$u_{gi,z} = -k_{gz} z_i - k_{dz} \dot{z}_i$

$u_{gi,x} = -si, x u_{gi,z} sgn(\dot{x}_i) - \mu_v \dot{x}_i$

$s_{i,x} = (\mu_c - (\mu_c - \mu_s)\exp(\frac{-|\dot{x}_i|^2}{v_s^2})$

$x_i$ and $z_i$ are the foot positions on the X and Z axes for foot $i$, $k_{gz}, k_{dz}$ are the spring and damping coefficients respectively, $u_{gi,x}, u_{gi,y}$ are the ground friction forces in X and Y, $\mu_c, \mu_s, \mu_v$ are the Coulomb, static and viscous friction coefficients respectively.

## D. Explicit Reference Governor

The ROM is modeled as a Triangular Inverted Pendulum (TIP). The two contact points at the feet allow us to estimate the distribution of the GRFs between the two legs so that we can evaluate the friction constraint on each leg individually. The kinematic constraints are modeled as

$$\ddot{p}_{fi} = \ddot{d}_{fi} + \ddot{p}_B = 0 \tag{11}$$

where $p_{fi}$ is the foot position, and $d_{fi}$ is the leg position from the body CoM. Assuming no slippage at foot endpoints, and both feet are on the ground, we have

$$\ddot{p}_{f1} = \ddot{p}_{f2} = 0 \tag{12}$$

and the body can be controlled using leg length acceleration

$$\ddot{d}_{f1} = \ddot{d}_{f2} = -\ddot{p}_B \tag{13}$$

so then the ground forces can be derived from the body acceleration as follows

$$m_B \ddot{p}_B = m_B g + u_{g1} + u_{g2} \tag{14}$$

It is said that this model restricts moment about the axis perpendicular to the support polygon and by assuming lateral ground forces are distributed evenly to form the constraint equations. So we end up with the general form equation

$$A[u_{g1}^T, u_{g2}^T]^T = b \tag{15}$$

and GRF can be solved given $A$ is invertible. ERG works as an add-on scheme to a pre-stabilized closed-loop system, manipulating the applied reference to the controller to enforce desired constraints while being as close as possible to desired reference trajectory.

## E. Constraint Equations

Some constraint equations are defined to enforce the no-slip condition on both the stance and the incline feet. Let the constraint equation be defined as

$$h_r(x_p, x_r) = J_r(x_p)x_r + d_r(x_p) \geq 0 \tag{16}$$

where the relationship between the constraint and $x_r$ is linear. The GRF for both feet can be solved as follows:

$$\begin{bmatrix} u_{g1} \\ u_{g2} \end{bmatrix} = A^{-1} \begin{bmatrix} m_B K(x_p - x_r) - m_B g \\ 0_{3x1} \end{bmatrix} \tag{17}$$

The following GRF constraints are given

$$|u_{gi,x}| \leq \mu_s u_{gi,z} \tag{18}$$

$$|u_{gi,y}| \leq \mu_s u_{gi,z} \tag{19}$$

which states that the absolute force on and planes must be less than the frictional force to prevent slippage

$$u_{gi,z} \geq u_{gi,z}^{min} \tag{20}$$

which states that the normal force must be greater than the minimum value of the friction pyramid surrounding each foot

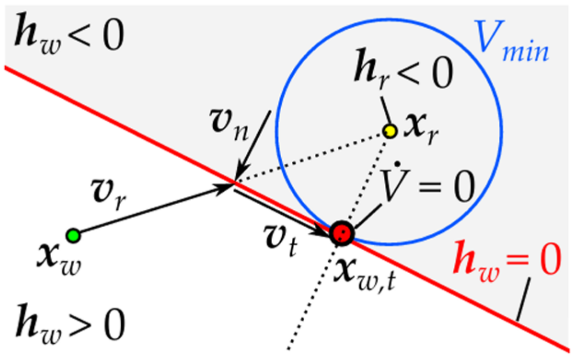


Fig. 1. ERG manipulates the applied reference states $x_w$ to be as close as possible to the desired reference $x_r$ without violating the constraint equation $h_w \geq 0$

to work. The way the ERG works is shown in the diagram below.

Consider the Lyapunov equation

$$V = (x_r - x_w)^T P(x_r - x_w) \tag{21}$$

$x_w$ is updated through the following law

$$\dot{x}_w = v_r + v_t + v_n \tag{22}$$

where $v_r$ drives $x_w$ directly to $x_r$, while $v_t$ and $v_n$ drives $x_w$ along the surface and into the boundary $h_w = 0$. The objective of ERG is to drive $x_w$ to state $x_{w,t}$ which is the minimum energy solution $V_{min}$ that satisfies the constraint $h_w \geq 0$.

The gradient $\dot{V} = 0$ if $\min(h_w) \leq 0$. While $\dot{V} < 0$ when $\min(h_r) \geq 0$ or when $\min(h_w) \leq 0$. In case both applied reference and target constraint equations are violated, then we have $\dot{V} > 0$ which drives $x_w$ towards constraint boundary shown in the image above. This allows $x_w$ to converge to $x_{w,t}$ which is the minimum energy solution that satisfies $h_w \geq 0$.

### F. Runge-Kutta Integration

The literature uses Runge-Kutta integration for the state update. The process works by taking 4 different integrals at 4 different points, and averaging them together to provide a more accurate estimate of the robot states. Only the first-order ordinary differential equations can be solved by using the RK4 method. The formula is given as a 5 part equation

$$K_1 = d(x, u, p) \tag{23}$$

$$K_2 = d(x + K_1\frac{dt}{2}, u, p) \tag{24}$$

$$K_3 = d(x + K_2\frac{dt}{2}, u, p) \tag{25}$$

$$K_4 = d(x + K_3 dt, u, p) \tag{26}$$

$$x_n = x + (\frac{K_1}{6} + \frac{K_2}{3} + \frac{K_3}{3} + \frac{K_4}{6})dt \tag{27}$$

where $x$ is the current state vector, $K_i$ are the intermediate dynamics results from the dynamics function $d$ (in our case the Husky model), and $dt$ is the step interval in simulation.

## III. Proposed Solution

We plan to make the ROM for Husky more accurate. This will allow us to implement more accurate dynamics equations and model the ERG to work on a model that is closest to the actual robot. This will allow us to analyze how accurately the presented literature will work on the physical robot. The big part of designing these paradigms is the sim-to-real transfer, which commonly never works because the properties of the robot were not created correctly, leading to incorrect dynamics equations. The figure below shows a picture of the real robot.

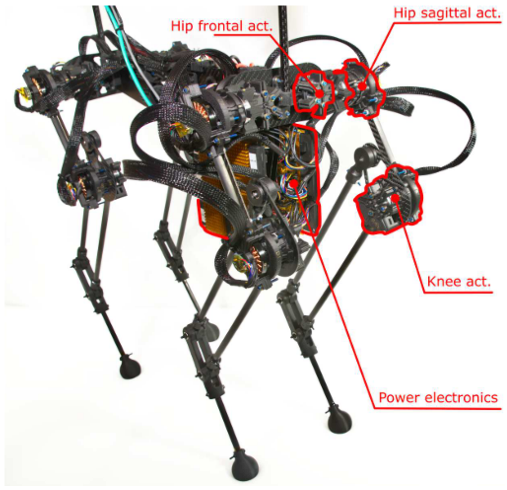


Fig. 2. Husky Carbon

The accuracy we measure will be shown through plots of body COM position and velocity, joint angles and velocities, and foot positions as the robot moves for a set time. We introduce some constraints to our system to create a stable testing environment, and to make sure we have good results for the time constraint of this project. Screenshots taken from a generated animation of the simulation show the GRFs acting on the robot, as well as how well the ROM balances itself. If the ROM is able to balance well and walk, we have verified that the Lyapunov stability constraints were satisfied and that this model will work on the physical robot well under our specified constraints. The project will show the quadruped walking at a reasonable commanded speed without generating any unreachable positions causing it to destabilize.

All simulation is done using MATLAB and some toolboxes (Symbolic Math, Control Systems, Sensor Fusion and Tracking, Curve Fitting, etc).

### A. Updated Joint Dynamics

Each leg now has extra joints for the knee, ankles, and thrusters (not used). We model this using variables in the robot state, denoted as follows:

$L_1$: Body to frontal hip
$L_2$: Frontal hip to sagittal hip
$L_3$: Sagittal to knee
$L_4$: Knee to ankle
$L_5$: Ankle to foot

$L_t$: Body to thruster

for each of the four legs $BR, BL, FR, FL$.

So then the position of the hip from the base is given by

$$p_{hi} = p_B + R_B L_1 i + R_B R_x(\theta_{hfi}) L_2 i \tag{28}$$

here, we take the rotation angle of the frontal hip and multiply that with our length of the segment to get to the sagittal hip position. From here, the position of the knee from the hip is

$$p_{ki} = p_{hi} + R_B R_x(\theta_{hfi}) R_y(\theta_{hsi}) L_{3i} \tag{29}$$

here, we add a rotation angle of the knee and multiply that with the length segment to get to the knee from sagittal hip. Similarly, the ankle position is

$$p_{ai} = p_{ki} + R_B R_x(\theta_{hfi}) R_y(\theta_{hsi}) R_y(\pi - \theta_{ki}) L_{4i} \tag{30}$$

where there is now a subtract from $\pi$ because the ankle mimics the knee and moves in the exact opposite direction. Finally, we have the ankle to foot transform, given by

$$p_{fi} = p_{ai} + R_B R_x(\theta_{hfi}) R_y(\theta_{hsi}) R_y(\pi - \theta_{ki}) R_y(\pi + \theta_{ki}) L_{5i} \tag{31}$$

which follows the same way but since the foot and knee are on the same movement axes, we add back the inversion we applied to the ankle. This gives us the complete system position dynamics with the new joints for each leg $i$ with respect to the base frame of the robot $B$. All of these values are entered into a struct named $p$ in the code.

In the simulation, we still use the same pre-generated functions provided by the literature to model our ground contacts and forces. The major changes implemented are in the mapping model for Husky Carbon, and the simulation to use the updated joints.

## B. Updated Lagrangian

The kinetic energy of the robot is a combination of all the kinetic energies associated with each joint and link. We take the difference in positions for each link in our time simulation to obtain the velocities for this calculation.

The potential energy is then the mass of each component multiplied by their position in the Z axis. Again, this is a linear combination, just like the kinetic energy.

Therefore, our Lagrangian $L$ can be constructed by taking the difference between the sum of linear and angular kinetic energies and the potential energy, denoted by

$$L = (K_l + K_a) - P \tag{32}$$

where, as an example, one component of the kinetic energy is given by

$$K_{ki} = \frac{1}{2} m_{ki} v_{ki}^2 \tag{33}$$

where the velocity is the difference in positions for the given joint between two time steps.

## C. Updated Simulation Code

With the new system dynamics and mappings in place, we now move into the simulation code. The rough structure is to iterate over $N$ timesteps, and at each step, take a step forward.

When taking the step forward, we use the provided Bezier function to generate our trajectory, and Inverse Kinematics to calculate the joint angles for the robot. Some parameters are listed below.

Gait Period = 0.2 seconds
dt = 0.0005 seconds
Simulation Time = 2.6 seconds
Step Length: 0.05 meters

We define the new state vector of size 50 as follows:

`x(1:8)`: Frontal and Sagittal hip joint values [FR, FL, BR, BL] (rads)
`x(9:11)`: COM position (XYZ)
`x(12-19)`: Frontal and Sagittal joint velocities [FR, FL, BR, BL] (rads/s)
`x(20-22)`: Base Linear velocity (XYZ)
`x(23-25)`: Base Angular velocity (RPY)
`x(26-34)`: Base rotation matrix (3x3)
`x(35-38)`: Knee joint values [FR, FL, BR, BL] (rads)
`x(39-42)`: COM velocity (XYZ/s)
`x(43-50)`: LuGre friction model

Our robot parameters for the Husky model consist of the following:

Body mass `p.mb`
Hip mass `p.mh`
Knee mass `p.mk`
Gravity constant `p.g`
Inertia of the base `p.Ib`
Inertia matrix of the COM `p.Icxx, p.Icyy, p.Iczz`
Body to hip frontal length `p.L1**`
Hip frontal to hip sagittal length `p.L2**`
Hip sagittal to knee length `p.L3**`
Knee to ankle length `p.l4*`
Ground constant `p.kb`
Body to thruster joints `p.Lt**`

We set the initial foot positions from the base in XYZ as `[0.2, 0.151, -0.6]`, `[0.2, -0.151, -0.6]`, `[-0.2, 0.151, -0.6]`, and `[-0.2, -0.151, -0.6]` for each of the legs. We then calculate the joint positions up the robot using IK.

Now, the simulation loop starts. For every time step, we first calculate the roll, pitch, and yaw of the base in the base frame. If we see that the norm of roll and pitch is greater than $\frac{\pi}{3}$, then the robot has tipped over. We then calculate the rate of change for the roll, pitch, and yaw to use in our PID system.

Next, based on which feet are in stance state and swing state, we calculate position of the feet in the robot base frame using the rotation matrix. We then use a bezier function to create the trajectory to the reference from where the current position is, and use IK to command the joints to that reference. Our PID control P section is defined as

$$u_P = K_p(Xj_{des} - Xj) \tag{34}$$

and our D control is

$$u_D = \frac{K_d((XJ_{des} - Xj) - Xe)}{dt} \tag{35}$$

and our I control is

$$u_I = K_i(Xe + (Xj_{des} - Xj)dt \tag{36}$$

where $Xe$ is the difference in desired joint states and actual states, i.e $Xj_{des} - Xj$, $Xj$ is the joint positions, and $dt$ is the step.

We pass this new control input to the Husky model. This will then update the robot state vector $x$ defined in the Lagrangian equations of motion. This will update the GRFs for that step, and also give back $\frac{dx}{dt}$. After this step, we use the ERG to meet our constraints of no angular components and only linear walking components to update our reference positions for the feet. In this simulation, we follow the same control scheme as the old simulation, so only the frontal and sagittal hips are actuated. The knee joint stays un-actuated but we record its values. We use RK4 integration to update the robot state vector.

## IV. Results

Both results use the same ERG constraints of only linear components.

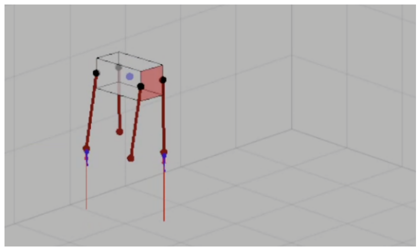
### A. Old Simulation Results

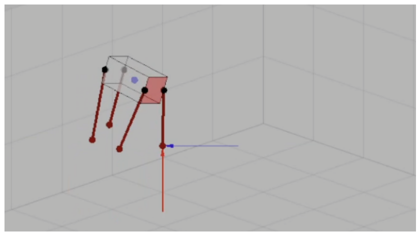
The old simulation results shown in Fig. 3. The ROM begins moving in the increasing x direction with the diagonal legs moving synchronously. As it moves forward, the body begins to sink forward due to lack of support from the front two legs, shown as drift in orientation. As the simulation progresses, the body starts to fall on the pitch axis, likely due to poor gain tuning. Additionally, as seen in the figures, the legs are straight stick-like legs with no knee or ankle joints. This greatly diminishes the accuracy of the model to the real world.

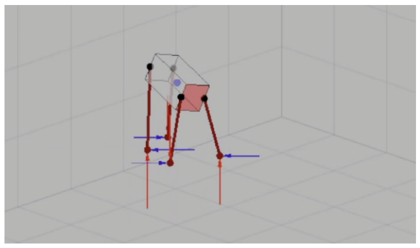
In Fig. 4 the plots describing the motion of the simulation are highlighted. In Fig. 4.a, the x, y, and z positions of each foot as well as their velocities in each direction are shown. These values are as expected, with the front and rear feet having a small difference in their positions but the velocities being similar.

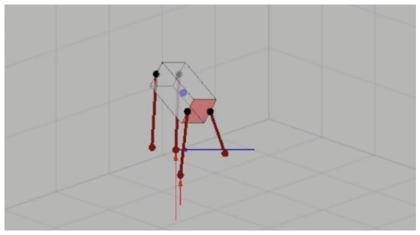
In Fig. 4.b, the position, velocity, orientation and angular velocity of the full body are shown. It is important to note that the in the body orientation plot, the roll, yaw, and pitch show the body's rotation about the 3 different axes. We would expect these values to stay consistent throughout the simulation, indicating a consistent orientation. However, the roll, pitch, and yaw are not consistent throughout the simulation. The roll, specifically, describes the rotation in the front-to-back axis. Since it is growing negatively, it shows that the body is rotating forward slowly, which is not ideal. Additionally, the yaw and pitch (rotation about the remaining 2 axes) are also not consistent. This may be a result of walking on stick-like legs, as they have no joints, rotate the body left and right in order to move forward.

Lastly, Fig 4.c describes the angles and velocity of the legs as well as the length of the legs. It is important to note that in the prismatic leg length plot, the length of the front two legs decreases minutely, while the length of the back two legs increases to compensate for lack of support from the front legs. This is unrealistic as in a real-life example, the length of the legs would not be changing and this what is fixed in the updated version of the simulation.


(a)


(b)


(c)


(d)

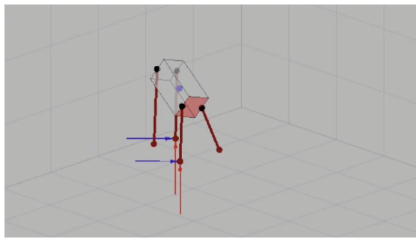
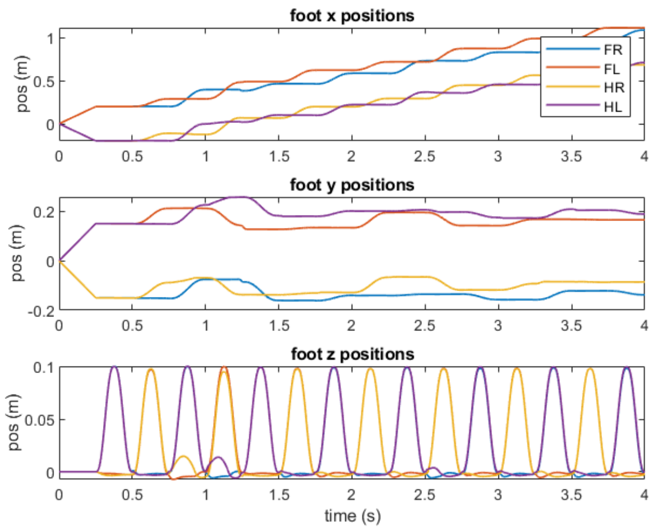
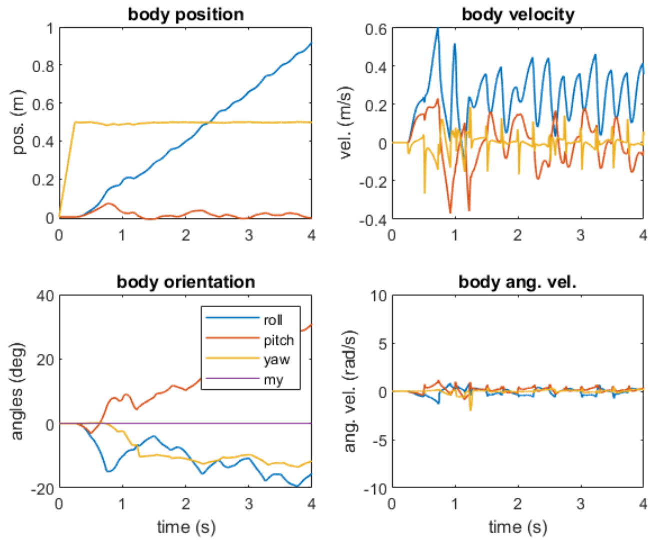

(e)

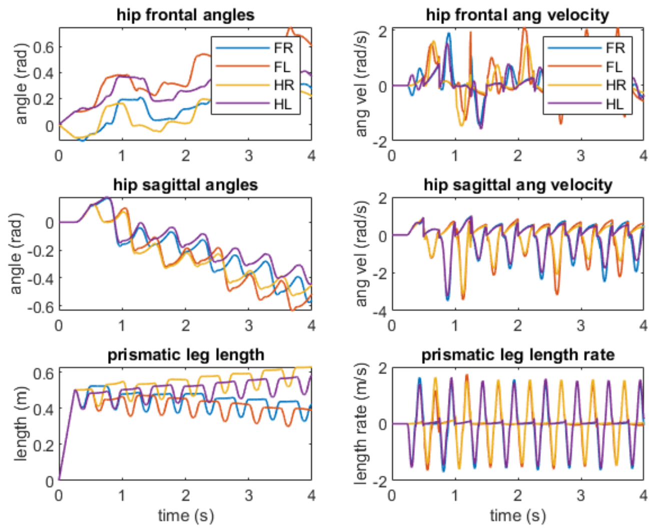
Fig. 3. Old Husky Carbon simulation using the ERG, shown with GRFs



(a)

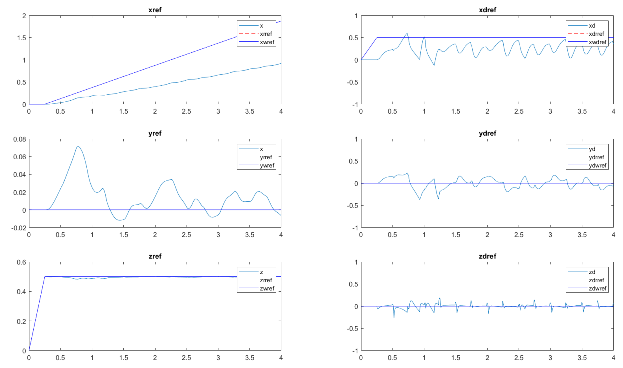


(b)



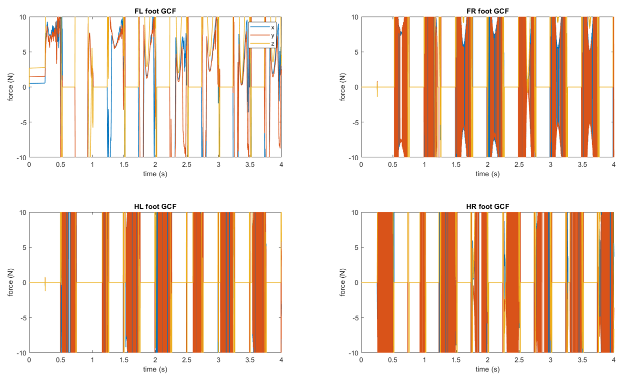
(c)

Fig. 4. Old Husky Carbon simulation Plots



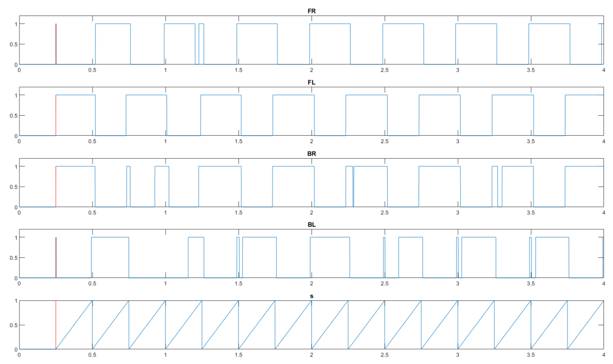
(a)

Fig. 5. Old Husky Carbon simulation Actual vs Ref XYZ



(a)

Fig. 6. Old Husky Carbon simulation Foot GCF



(a)

Fig. 7. Old Husky Carbon Simulation Leg States

### B. New Simulation Results

The new simulation results shown in Fig. 8. show that the Husky Carbon performs a lot better, and the ROM is much more accurate than the old one. As seen in the figures, the body is now supported by legs that have multiple joints, making the movement more seamless and reducing the change in an upward support for the body, which results in no forward sinking of the body. Visualizing the GRFs show that the model is able to prevent slipping and walk a stable gait from start to end in a straight line. In the scope of this experiment, we wish to only test straight line walking.

In the new plots showing foot positions (Fig. 9.a), we see that the x positions are all rising together, indicating forward motion. The y positions are roughly stable, with minimal oscillation in the lateral direction, indicating there is no lateral movement. This means the ERG is enforcing the linear movement only constraints. The z positions oscillate as they should, indicating the feet are moving up and down.

The body COM position (Fig. 9.b) shows only the x position linearly increasing, and the orientation and angular velocities staying 0, again indicating our ERG is working as it should.

The frontal hip angle has little deviation, and remains roughly at 0 radians. The sagittal hip angle plot is shown to oscillate back and forth, indicating the forward motion. Notice that two of the legs are exactly out of phase with the other two, indicating a stable gait (Fig. 9.a).

The rest of the plots show the constraints (Fig. 10.), where $x$ is what we applied and $xr_{ref}$ is the constraint to meet. Since our command is over that value the ERG does not need to correct for it.

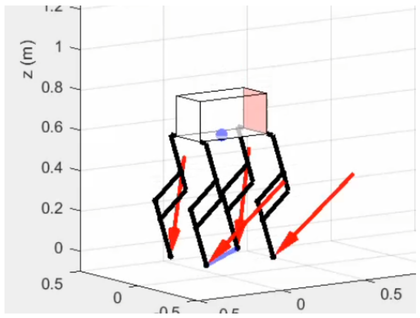
The GCFs (Fig. 11.) for each foot show the force recorded in Newtons as Husky makes contact with the ground. In x, we see that the applied reference already satisfies constraints, so it is not adapted. In y, we see it is basically 0, but since it is negative ERG acts to correct it to 0. In z, we are below 0.7, so ERG corrects the applied references to 0.7 height

The leg states for the hip and knee are shown in Fig. 12. We have run the simulation for 20 seconds to show that the system is indeed stable for a long time, and that the ERG is acting to enforce our linear movement only constraints. Overall, the performance of the updated simulation is significantly greater than the performance of the old simulation.

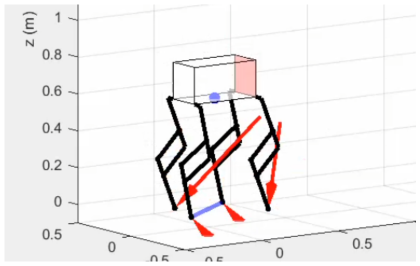
## V. Conclusion

In this project, we have learned about common challenges presented in the field of legged locomotion, and we have analyzed literature on an optimization-free approach. We have used concepts like Lyapunov stability constraints to understand how the equations are derived and work, as well as updated the MATLAB code provided by the authors of the literature to include a more updated version of the robot. We see that the updated ROM performs a lot better compared to the initial ROM, and so the sim-to-real gap can be bridged using this work.

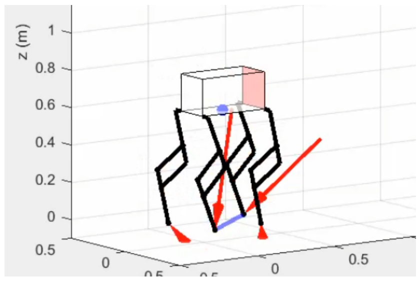
For the future, the constraints can be loosened to allow angular components too, which will produce a better estimate



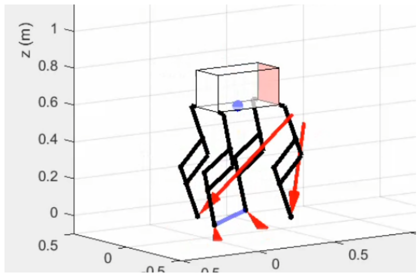
(a)



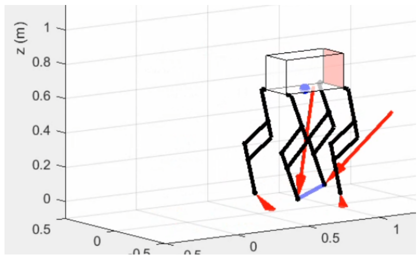
(b)



(c)

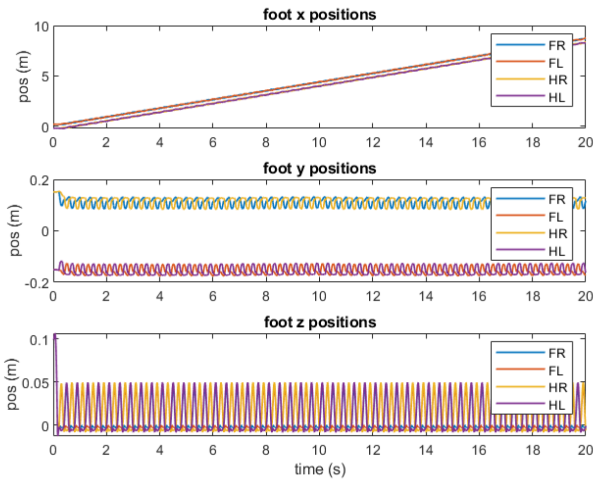


(d)

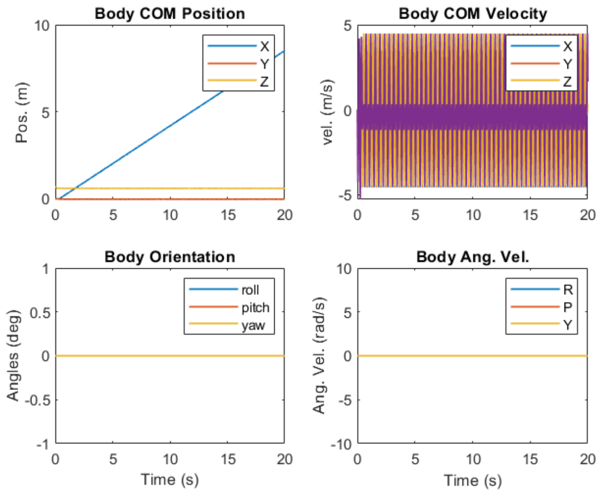


(e)

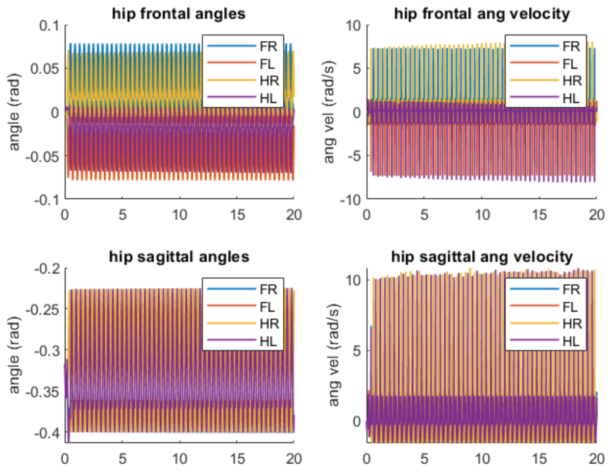
Fig. 8. New Husky Carbon simulation using the ERG, shown with GRFs



(a)

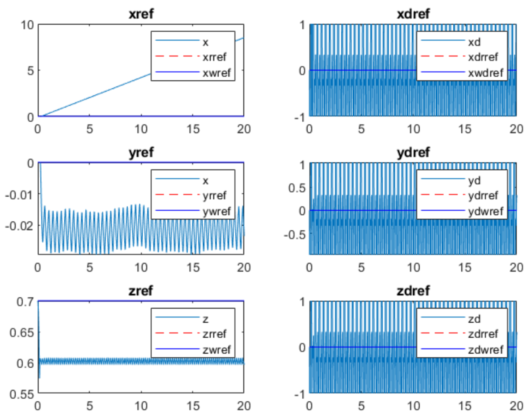


(b)



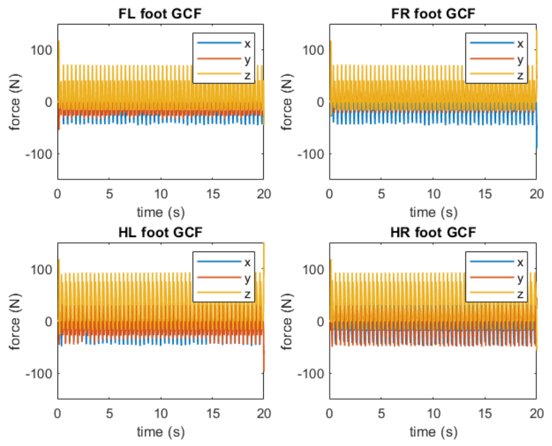
(c)

Fig. 9. New Husky Carbon simulation plots, showing joint angles and velocity, body CoM position and velocity, and foot positions in XYZ



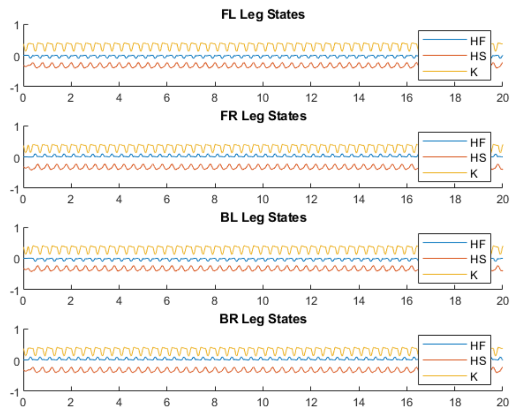
(a)

Fig. 10. New Husky Carbon simulation showing the ERG corrections being made to satisfy linear constraints



(a)

Fig. 11. New Husky Carbon simulation Foot GCF in XYZ



(a)

Fig. 12. New Husky Carbon Simulation Leg States

of the real world. Right now, choosing to do that makes the system unstable, because the thrusters on Husky are not modeled yet. Husky uses these thrusters to balance itself against angular velocities that are unnecessary. This allows it to traverse terrain much faster and also perform narrow path walking. Future work will entail changing the model to incorporate the thrusters too.

## VI. Acknowledgements

We would like to thank the Ph.D. students of the Silicon Synapse Lab for providing us with the code necessary for this work and for helping us understand parts of it as we evolved it to the updated version.

## References

[1] Eric Sihite, Pravin Dangol, and Alireza Ramezani. "Optimization-free Ground Contact Force Constraint Satisfaction in Quadrupedal Locomotion". In: *CoRR* abs/2111.12557 (2021). arXiv: 2111.12557. URL: https://arxiv.org/abs/2111.12557.