Optimization-based control for Biped Jump

Anirudh Kailaje
University of Pennsylvania
kailaje@seas.upenn.edu

Dhruv Parikh
University of Pennsylvania
dhruvkp@seas.upenn.edu

***Abstract*—We present a 3-phase hybrid controller that jumps to a required height. The launch trajectory is obtained through contact implicit Direct Collocation, a mathematical program with complimentary constraints (MPCC). The trajectory is followed by an Operational Space Controller (OSC). Separate OSCs were made for the flight and landing phases. The controller we made is robust against a wide range of system mass fluctuations, state estimation errors, and terrain geometry. The controller works for a range of jump height requirements for a given system mass distribution, estimation error, and terrain. It is sensitive to low coefficient of friction, solver parameters, and phase switch error when direct collocation fails.**

I. Supplementary Material

All experiments are reproducible and all logs are available: Code: [*Github*] | Slides/Logs: [*Google Drive*]

II. Introduction

While walking and running are essential aspects of robotic mobility, the ability to perform controlled jumps adds a new dimension of agility. This capability enables the exploration of environments with elevation changes or obstacles. A jump motion involves the robot reaching a specific upward velocity, resulting in flight and landing. The complexity of the problem arises from multiple contact modes and actuation authority.

This paper provides a framework for making a planar bipedal robot jump to a required height. Our control framework utilizes whole-body dynamics and addresses the problem through nonlinear optimization. It eliminates reduced-order models that may not accurately represent the robot's connection to its structure and morphology. The proposed architecture has succeeded across various terrain conditions, estimations, and dynamics uncertainties. The controller is designed exclusively for precise jumping and stable landings and doesn't support general locomotion tasks such as walking or running with diverse gaits.

We begin by solving the problem of reaching initial velocity using direct collocation. The next step is to formulate a lower-level control for reference tracking. This is done by an Operation Space Quadratic Program (OSQP) on the whole body. Once in the air, the robot has no control authority; however, we must ensure that we land in a stable and recoverable condition. This necessitates an in-flight controller and a landing controller for regulation. We describe the entire framework in detail in the subsequent sections.

III. Background and Related Work

As demonstrated in the 90s by Hodgins and Raibert, under-actuated robots showcased acrobatic maneuvers using simple models [1]. The focus was on carefully designed objectives and PD control for tracking. Using open-loop actuator signals, they achieved a stable somersault in a planar biped with a 90% success rate. However, these methods require precise initial conditions and state estimates, and landing stability post-maneuver was questionable [2].

In the modern optimal control literature, Xiong and Ames worked on reduced order model control on Cassie to make it jump [3] and Somersault [4]. Their approach was to approximate the full-order Cassie to a simple model, use the direct transcription method to solve the trajectory optimization problem, and then track using OSQP. The parameters of the reduced order model were obtained using parametric regression. Another variant of such an approach is in [5], where the flight phase joint trajectory is obtained in a closed-form solution. However, this control won't be robust to model uncertainty.

We believe jumping and somersaults are the same paradigms of problems. Both use a trajectory optimization framework to get the biped to a desired condition such that it executes a maneuver during free flight and then does a stable landing. The constraints in our controller draw inspiration from both of these problems, resulting in a better-guided solution of nonlinear optimization. We attempt to view this problem from an engineering lens by modularizing the control hierarchy and presenting which control strategy in each module works better than the other.

IV. Approach

A. Control Architecture

As discussed in III, we use offline nonlinear trajectory optimization to generate a state space trajectory that jumps at the final knot point. The Finite State Machine (FSM) feeds the relevant trajectory to a lower-level OSQP based on the state and time conditions. Based on the mode, OSQP tracks the relevant objective. Figure 1 describes a simplified control hierarchy.

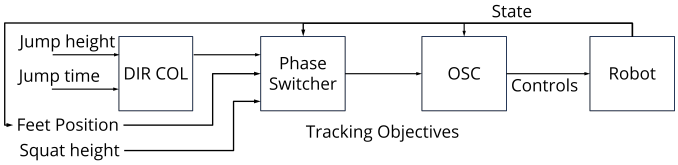

Fig. 1: Controller Block Diagram

B. Finite State Machine

The finite state machine takes the clock time, feet position, and the center of mass velocity to determine the phase. The phases are defined according to the parameters in table I:

Phase	Context Time	Feet Position	COM Velocity
Launch	$< 1.1 \cdot$ Jump Time	$< 10^{-2}$	
Flight	$> 0.9 \cdot$ Jump Time	> 0	$> 0.9 * \sqrt{2 \cdot g \cdot \text{Jump Height}}$
Landing	$> 1.1 \cdot$ Jump Time	$< 10^{-2}$, either leg	< 0

TABLE I: Definition of Controller Phases

C. Trajectory Optimization

Posa et al. proposed a contact implicit trajectory optimization, called Constrained Direct Collocation (DIRCON) [6], where the trajectory optimization problem is written as an optimization with Complimentary Constraints on foot position and friction cone. We based our direct collocation on the same. The costs were applied to control actions. For brevity, we have skipped mentioning the collocation constraints.

Firstly, we constrain the first pose as the initial state. The final vertical velocity is to be within a specified tolerance of the required velocity, and the final angular momentum of the torso about the hip $L(x_k, u_k)$ to be within a specified limit, and the change in Z velocity to be within specified limits. Equation 1 describes the above-specified constraints.

$$\begin{aligned} x_0 &= \text{initial state} \\ v_{des} - v_{tol} \leq \dot{x}_{N,z} &\leq v_{des} + v_{tol} \\ -L_{tol} \leq L_k(x_k, u_k) &\leq L_{tol}, k \in [N-3, N] \\ -20 \leq \dot{x}_{k,z} &\leq 9 \end{aligned} \tag{1}$$

Equation 2 forms the complementary constraints that describe the impacts (assumed to be inelastic) and friction effects with coefficient μ [6]. They ensure that the forces are within the friction cone if the contact is in a static friction regime. If sliding, the force is at the limit of the friction cone and in the direction opposite to the tangential velocity of the contact.

$$\begin{aligned} \phi(x_k), \lambda_{k,z}, \lambda^+_{k,x}, \lambda^-_{k,x}, \gamma_k &\geq 0 \\ \mu\lambda_{k,z} - \lambda^+_{k,x} - \lambda^-_{k,x} &\geq 0 \\ \gamma_k \pm \psi(x_k, \dot{x}_k) &\geq 0 \\ \phi(x_k)^T \lambda_{k,z} &= 0 \\ \left(\mu\lambda_{k,z} - \lambda^+_{k,x} - \lambda^-_{k,x}\right)^T \gamma_k &= 0 \\ (\gamma_k + \psi(x_k, \dot{x}_k))^T \lambda^+_{k,x} &= 0 \\ (\gamma_k - \psi(x_k, \dot{x}_k))^T \lambda^-_{k,x} &= 0. \end{aligned} \tag{2}$$

Additionally, since drake [7] uses 3D descriptions, we had to keep the contact forces defined as $\lambda_k = [\lambda^+_{k,x} - \lambda^-_{k,x}, \lambda_{k,y}, \lambda_{k,z}]$ and constrained $\lambda_{k,y}$ as zero. We also kept the X velocity close to zero, which can be changed as required to gain a forward jump. We also added constraints to limit the joint angles within joint limits and controls within control limits. Equation 3 defines the constraints discussed above.

$$\begin{aligned} \lambda_{k,y} &= 0 \\ -0.01 \leq \dot{x}_{k,x} &\leq 0.01 \\ \text{lower joint limits} \leq q_k &\leq \text{upper joint limits} \\ -\tau_{max} \leq u_k &\leq \tau_{max} \end{aligned} \tag{3}$$

To guide the solution, we introduce additional constraints on the torso angle and base z velocity for the first $N-3$ knot points and put limits on the z position of the base as shown in equation 4

$$\begin{aligned} -0.01 \leq \theta_k &\leq 0.01 \\ -20 \leq \dot{x}_{k,z} &\leq 9 \text{ for } k \in [1, N-3] \\ -9 \leq \dot{x}_{k,z} &\leq 30 \text{ for } k \in [N-3, N] \\ 0.6 \leq x_{k,z} &\leq 1.2 \end{aligned} \tag{4}$$

D. Operational Space QP Control

The Operational Space Controller tracks various objectives depending on the finite state. The mathematical program is in equation 5:

$$\begin{aligned} \min_{\lambda,\ddot{x},u} \sum_{i=1}^{k} &(\ddot{y}_i^{cmd} - \ddot{y}_i)^T \cdot W_i^m \cdot (\ddot{y}_i^{cmd} - \ddot{y}_i) \\ \text{s.t } & M\ddot{x} + C\dot{x} = \tau_g + Bu + J_c\lambda \\ & J_c\dot{x} + J_c\ddot{x} = 0 \\ & |\lambda_x| \leq \mu\lambda_z \\ & \lambda_z > 0 \\ & |u| < 0.7\tau_{max} \end{aligned} \tag{5}$$

Here, m represents the mode, W_i represents the cost, k represents the tracking objective, λ represents the contact force, and τ represents joint torque. It is to be noted that in the event of flight, only the following constraints are active (eq. 6)

$$\begin{aligned} M\ddot{x} + C\dot{x} &= \tau_g + Bu \\ |u| &< 0.7\tau_{max} \end{aligned} \tag{6}$$

With the finite states, the tracking objective ($\ddot{y}_i^{cmd}$) changes. Table II shows the tracking objective with respect to mode.

Mode (m)	COM	Foot	Torso
Launch	✓		✓
Flight		✓	
Landing	✓		✓

TABLE II: Tracking objectives with modes

The tracking objectives are generated using a PD controller. Hence, the performance of OSC highly depends on it requiring a careful design. A series of tests were performed to reach the specified structure. Each mode has different gains, with feedforward terms only active during the flight phase. We observed that instead of tracking velocity, the derivative of output variation of PD [8] avoided accidental overshoot. Equation 7 shows the PD control strategy.

$$\ddot{y}_i^{cmd} = \ddot{y}_{des} + K_p(y_{des} - y) + K_d(-\dot{y}) \tag{7}$$

E. Intricacies

This sub-section highlights certain intricacies in our code that made the system more robust.

1) PD Controller: In addition to eq 7, we also saturate the controller output as shown in equation 8.

$$|\ddot{y}_i^{cmd}| \leq c \tag{8}$$

Here, c is the saturation limit. This is useful in certain scenarios. For example, if we are in land mode, when increasing the COM height, c is set to cap acceleration commands at 1 G, preventing unintended biped relaunch during the land phase. Moreover, the feed-forward term in PD is only active during the launch phase. Since, the rest of the time, we only want to regulate the states, we don't want to risk instability for aggressive $\ddot{y}_i^{cmd}$.

2) Torso Error on Manifold: Control of angles is problematic when the angle wrapping around from the desired value to the current value does not take the shortest path. This was an issue because if the system overshoots, the torso rotates fully 360° to reach back to the 0°, and then it overshoots again,

resulting in circular motions. The solution is to calculate the error on 1D manifold as defined in equation 9:

$$e_\theta = \text{sgn}(v_1^T \times v_2) \cdot cos^{-1}(v_1^T v_2) \tag{9}$$

Here v_1 and v_2 are the current and desired torso rotated vector in $\mathbb{R}^2$, and e_θ is the error term for torso angle.

3) Warm Starts: We initialized OSQP's u in the launch phase with DIRCON control trajectory output. With this, the OSQP found solutions with lower control effort while still being able to track compared to the cold start.

In the flight and landing OSQPs, we used the previous solutions as warm starts for faster convergence.

4) Solver Parameters: We prioritized trajectory feasibility over optimality in our solver parameters in an attempt to get at least one feasible trajectory. Hence, our "Minor feasibility tolerance" is lower than the "Major Optimality tolerance."

5) Solver Fail Handle: In case of a failure in OSQP, we set the solution to the previous one. Moreover, in the event of a solution that is greater than the actuation limit, we reduce the magnitude by the following equation:

$$u_i = 0.7\tau_{max} \cdot \frac{u_i}{||u_i||_2}$$

F. Experimental Setup

The robot chosen was a planar walker biped with a torso and two upper and lower leg links. The robot employs torque-actuated hip and knee joints for control. The simulation and control were done using PyDrake [7]. The environment comprised a ground plane with varying slopes, friction, and sometimes boxes to model discontinuity. We developed a reliable logging structure to run more than 1000 experiment trials and maintain reproducibility. Our logs recorded DIRCON outputs, states, tracking data, and visualizations, which significantly expedited the extraction of insights from diverse experimental scenarios.

V. Experiments

A. Subsystem Testing

To avoid coupling errors of multiple modes, we performed subsystem testing where each subsystem's role in success was defined.

1) Direct Collocation: Direct Collocation's role is to provide a feasible trajectory that only makes the robot jump at the last knot point to reach the desired height. We had a lower level OSC tracking DIRCON trajectory and only simulated till the flight phase with no flight controller. During the development of the module, we wanted to test the following things about the subsystem:

- *Is the mathematical program properly formulated?*
  We validated the collocation mathematical program by testing a simplified version, ensuring correct construction and implementation of constraints for maintaining the initialized pose.
- *Does the direct collocation work for simpler tasks?*
  After confirming the program's ability to maintain a pose, we introduced additional constraints, including a test where the biped was instructed to squat.
- *Does it work for jumping?*
  When introducing tasks like squatting and jumping, we consistently observed program failures or infeasible constraints, signaling a need for correction in handling friction cone constraints, leading us to address complementary constraints.

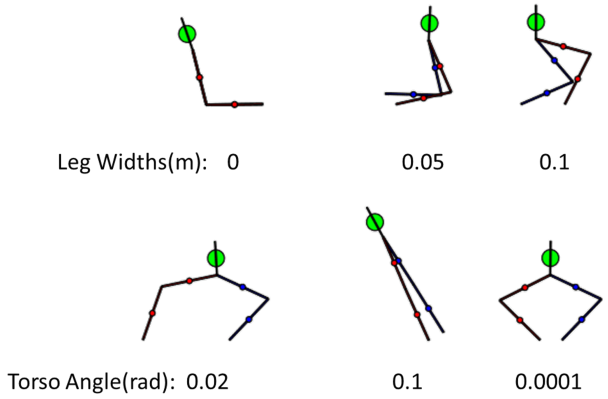

Fig. 2: Stress testing for landing

2) Flight and Landing: For flight, we don't have control authority to influence the center of mass. Initially, there were no plans for in-flight control itself. We performed drop tests by varying torso angle, height, cost structure, and leg width. Selected results are shown in fig 2. It was found that keeping width between legs with symmetry when landing provided tolerance for large torso angles. We also concluded that keeping quadratic cost on control effort made it difficult for a biped to stabilize.

With this, an in-flight controller was made, which tracks foot trajectory to keep a constant width between both feet. Later, the landing controller's job was to make the torso angle upright and reach a predefined COM that maintained the position.

B. Systems Testing

With the above analysis, we present the entire system's experimentation. Our aim is to characterize the performance of our system from a robustness perspective, particularly regarding uncertainty. Our experiments are fully reproducible (Sec, I). Tables III, IV, V outline our controller characterization process.

1) Parameters: The control parameters of all experiments were kept constant and are described in table III:

Parameter	Subsystem	Value
Knot Points	Traj Opt	5
Jump Height	Traj Opt	0.5 m (relative)
Jump time	Traj Opt	0.5 s
Minor Feasibility Tolerance	Traj Opt	1e-5
Major Optimality Tolerance	Traj Opt	1e-3
Leg Width	Flight	0.2 m
COM K_p, K_d	Launch	60, 0
Torso K_p, K_d	Launch	0, 100
Foot K_p, K_d	Flight	1500, 30
COM K_p, K_d	Land	300, 100
Torso K_p, K_d	Land	5.85, 2

TABLE III: Parameters for experiments

2) Design of Experiments (DoE): Given these parameters, we identified the objectives as defined in Table V that we wanted to evaluate against. For these objectives, we identified the factors that affect the outcomes and tested the controllers with factors at specified levels in Table IV.

3) Results: We ran over 50 trials cumulatively across all of the factors and their levels. Table V shows the summarized results. We observed the failures and have summarized the factors that our controller is robust against and the factors it is sensitive to, summarized in Table VI As a representation,

Parameter	Unit	Range
Torso Mass	kg	[0.8, 1.2] × Default Values
Leg Mass	kg	[0.5, 2] × Default Values
State Noise	-	$\{70\text{ cm}, 5\text{ rad}\}\ 3\sigma$
Terrain	deg	[-45,45]
Coefficient of friction (μ)	-	[0.1, 1.1]
Collocation Methods	-	DIRCOL or DIRCON

TABLE IV: DoE: Factors and Levels

Subsystem	Objectives	Possibilities	Results
Direct Collocation	Mathematical Program	Iteration Limit	100%
		Solver Specific Error	0%
		Infeasible Constraints	0%
		Solution Found	0%
	COM Tracking	Follow trajectories	84%
Flight	Foot Tracking	Follows trajectories	100%
	Torque Saturations	Min/Max Saturations	2%
Landing	Stabilize Post Landing	Doesn't Fall off or launch again	88%
Phase Switch	Correct Mode Switching	Yes/No	84%

TABLE V: Summary of Results (50 trials: logs)

Robust Against	Sensitive to
Mass fluctuations	Coefficient of friction
State Estimation Error	Solver Parameters
Terrain Geometry	Phase Switch Inaccuracy

TABLE VI: Factor robustness of controller

we show two experiments - one in an ideal scenario and the other with model, estimation, and terrain uncertainty combined. Figure 3 & 4 show the tracking plots for both. An example of stabilizing in uneven terrain is demonstrated in 5. Our controller, although not designed explicitly for these conditions, was able to perform a stable biped hop.

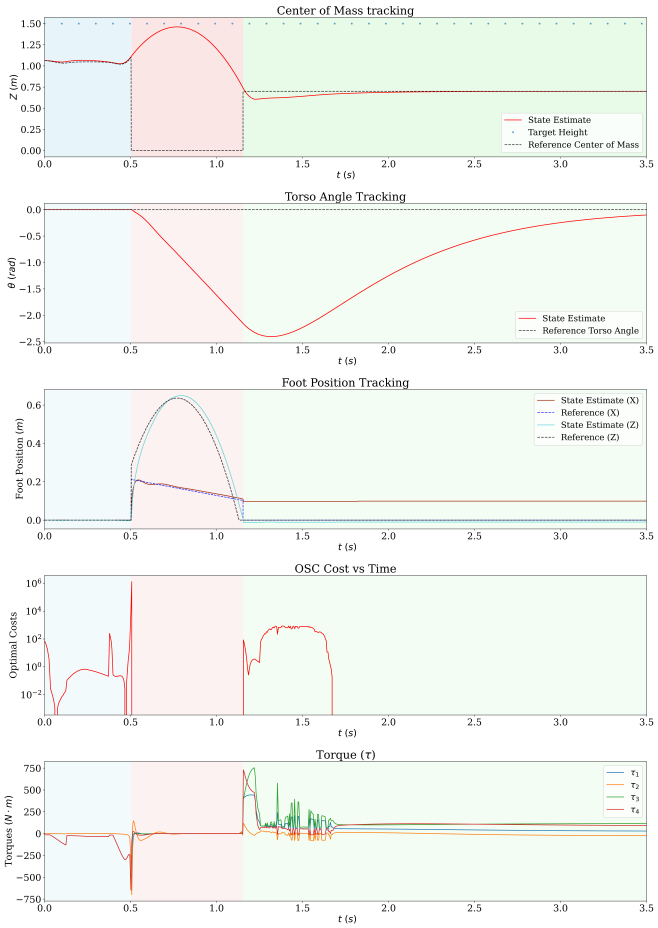

Fig. 3: Baseline Experiment

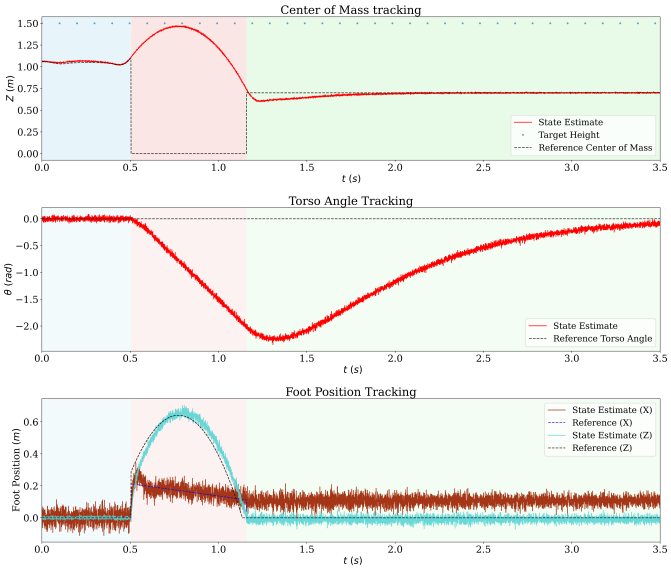

Fig. 4: Stress Experiment

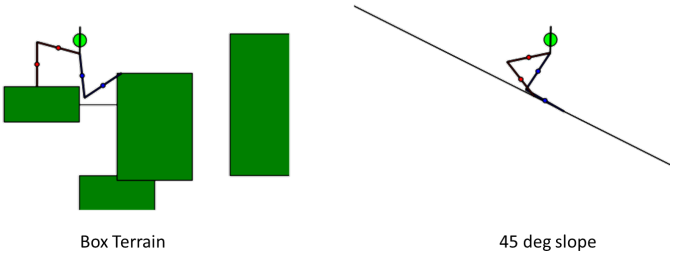

Fig. 5: Difficult Terrain Tests

VI. Discussion

The key findings with our controller are the following:

- The controller works for a family of jump heights, given a terrain (factors of slope, μ).
- The controller works across a large value of slopes and can stabilize post-landing without any changes.

During the experimentation, we noticed the following problems that call for a careful examination in future work:

1) Direct Collocation
   - Very sensitive to solver settings.
   - Lower cost "solutions" give infeasible trajectories.
2) Phase Switch can be more robust using contact forces.
3) Using complementary constraints in OSC should allow for slip.

VII. Conclusion

This work presents a robust control framework for facilitating jumps in a planar biped. The launch trajectory is determined using DIRCON, and a 3-phase Operational Space Controller (OSC) is employed to track the trajectory and stabilize the landing. Testing based on first principles demonstrates the framework's robustness to terrain, mass, and state uncertainty. Sensitivity is observed in relation to friction and solver parameters. The future objective is to investigate solver intricacies and extend the framework to 3D bipeds.

References

[1] J. Hodgins and M. H. Raibert, "Biped gymnastics," *Dynamically Stable Legged Locomotion*, vol. 79, 1988.

[2] R. R. Playter and M. H. Raibert, "Control of a biped somersault in 3d," in *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems*, vol. 1, pp. 582–589, IEEE, 1992.
[3] X. Xiong and A. D. Ames, "Bipedal hopping: Reduced-order model embedding via optimization-based control," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3821–3828, 2018.
[4] X. Xiong and A. D. Ames, "Sequential motion planning for bipedal somersault via flywheel slip and momentum transmission with task space control," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3510–3517, 2020.
[5] D. Tian, J. Gao, C. Liu, and X. Shi, "Simulation of upward jump control for one-legged robot based on qp optimization," *Sensors (Basel, Switzerland)*, vol. 21, no. 5, p. 1893, 2021.
[6] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 69–81, 2014.
[7] R. Tedrake and the Drake Development Team, "Drake: Model-based design and verification for robotics," 2019.
[8] MathWorks, "Pi-d and i-pd controllers." Accessed: 12, 2023.