Project Assignment 1
RAS 557 – Foldable Robotics

Group 12

Anusha Chatterjee , Abhijit Sinha and Shivakumar Sridhar

1. Project Goal and Fit

1.1 Project Goal

The objective of this project is to design, simulate, and experimentally evaluate a bio-inspired foldable robotic leg that is based on a four-bar linkage, mimicking the hind leg of a locust. The study will explore how varying length ratios between the femur and tibia segments affect jumping performance, focusing on jump height, horizontal distance, and mechanical efficiency.

1.2 Research Question

Primary Research Question

1. What impact do the length ratios of the femur and tibia segments within a four-bar, locust-inspired jumping mechanism have on jump height, jump distance, and mechanical efficiency?

1.3 Scope

The project involves developing a planar four-bar leg model inspired by the anatomy of a locust's femur and tibia. It includes kinematic and dynamic analysis of the mechanism using simulation tools such as MuJoCo and Python. A parametric study will be conducted, systematically varying the lengths of the femur and tibia (and their ratio) to assess their effects on:

- Maximum jump height
- Horizontal distance
- Energy or work required per jump (mechanical efficiency)

Additionally, the design and fabrication of a foldable physical prototype will be carried out using lightweight materials, including paper, cardboard, acrylic, or 3D-printed components. The prototype's jumping performance will be experimentally tested and compared with the simulation results.

1.4 Impact

Jumping is an essential capability for small robots that need to navigate uneven terrain, such as gaps, steps, and rubble. Actuation that mimics grasshoppers is particularly attractive because it allows for high peak power through the elastic storage and rapid release of energy. Advances in foldable fabrication and the use of flexible composites have made it feasible to design lightweight, compliant mechanisms that replicate this strategy on a small scale.

1.5 Team Fit

This project represents a strong alignment with interdisciplinary fields, integrating the principles of kinematics, mechanism design, and manufacturing constraints, all while drawing inspiration from biomechanics. The focus on "mechanical intelligence," which encompasses geometry, material compliance, and innovative latching mechanisms, moves beyond reliance on high-bandwidth actuation. This approach is particularly relevant to the emerging paradigm of foldable robotics, highlighting the potential for sophisticated, adaptable designs that prioritize efficiency and functional versatility.

1.6 Topic Fit

The leg is engineered as a foldable linkage featuring compliant joints or flexures that store strain energy during a gradual “cocking” phase. A latch secures the mechanism in this high-energy state until it is released, triggering a rapid extension event. This design effectively utilizes foldable robotics techniques, incorporating thin flexures for the joints, laminated links for enhanced stiffness, and a geometric layout that can be cut, laminated, and folded.

2. Background Research (Literature Review)

2.1 Search Strategy

In our investigation, we utilized Google Scholar to enhance our literature search by employing a strategic combination of keywords related to organism dynamics and mechanical principles. Specifically, we formulated queries such as “grasshopper biomechanics jump takeoff angle,” “locust take-off ground reaction forces,” “semi-lunar process resilin energy storage,” “bio-inspired jumping robot four-bar leg,” and “foldable mechanism compliant hinge.” This approach allowed us to gather a comprehensive array of research that bridges biological motion with mechanical systems, thereby informing our understanding of bio-inspired designs and their applications in robotics.

2.2 Key Figures from the Literature

Include at least two figures: one anatomy/structure figure and one kinematics/force/motion figure.

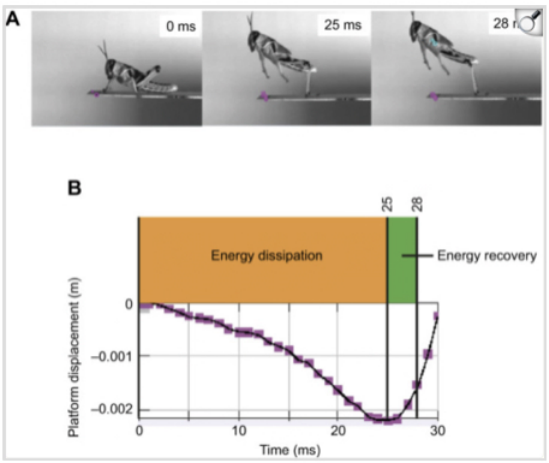


FIGURE: TAYLOR, DEEMING & SUTTON (2024, JEB) — PANELS SHOWING PLATFORM DISPLACEMENT VS. TIME AND JUMP KINEMATICS ACROSS SUBSTRATE STIFFNESS (THEIR ENERGY-RECOVERY EXPERIMENT).

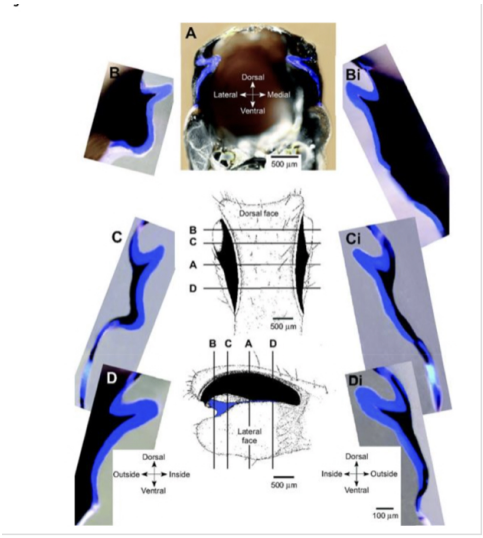
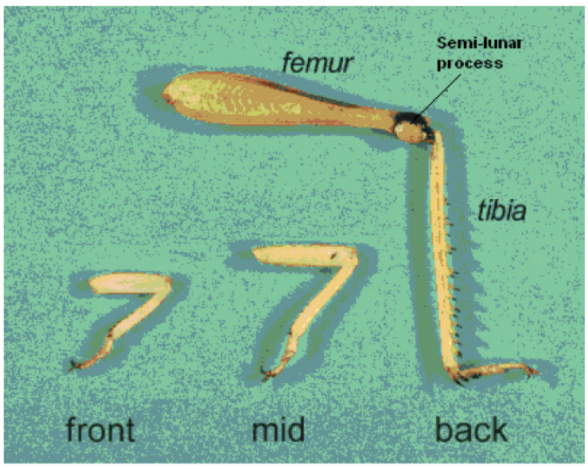


FIGURE 2- BURROWS & SUTTON (2012, JEB) — Micrographs and schematics of the semi-lunar process (SLP) illustrate the resilin wedge bonded to hard cuticle, supporting the composite-spring assumption in my model and inspiring the "variable stiffness" concept.



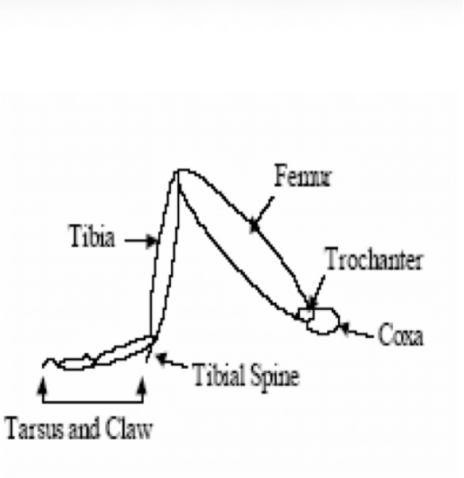


FIGURE 3:The hind leg with identified segments (Fauske, 2002; Laksanacharon, Pollack et al., 2005; Laksanacharon, Quinn, and Ritzmann, 2003).

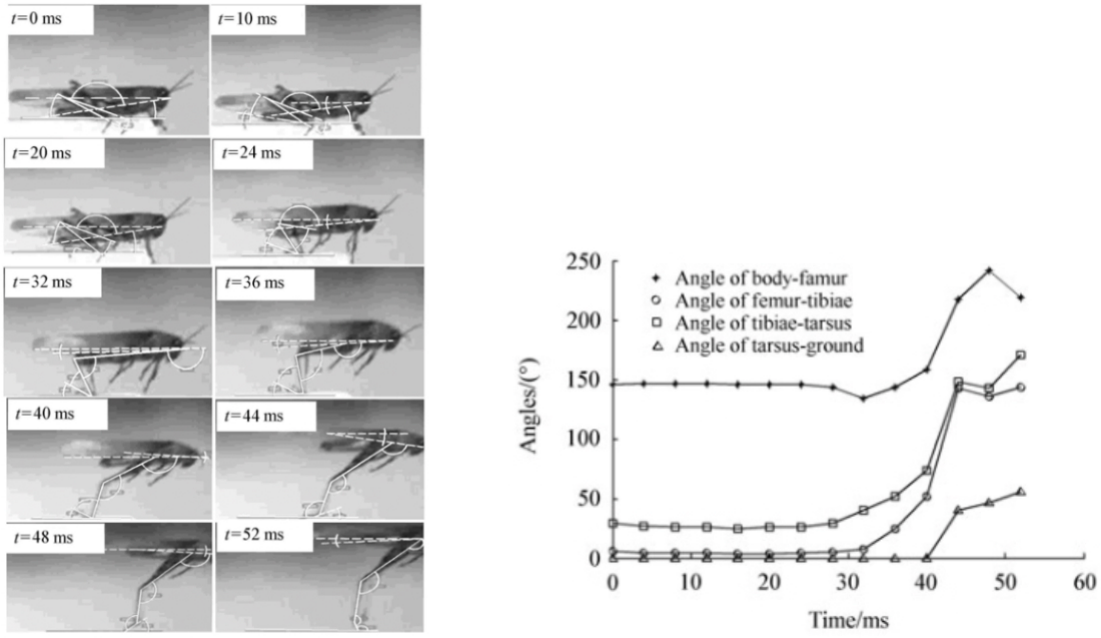


FIGURE 4: Chen et.al (2014):(a) A high-speed camera operating at 500 frames per second captures the take-off sequence of a locust over a duration of 0 to 52 milliseconds. The frames illustrate the progression from the pre-loading phase at t = 0 ms, through leg extension at approximately 40 ms, to the moment of airborne take-off around 52 ms.
(b) The angles between the body and femur, femur and tibia, tibia and tarsus, as well as tarsus and ground, are plotted as functions of time from 0 to 60 milliseconds.

2.3 Value of Each Paper (What We Use It For)

[1] Hawlena et al. (2011) — performance targets and takeoff geometry
This study presents direct, quantitative measurements of takeoff, specifically speed and launch angle, using motion tracking. These values are essential for establishing realistic performance goals for a bio-inspired jumper. Additionally, it emphasizes that the takeoff angle can vary depending on the context (such as escape versus distance), which reinforces the idea of considering launch angle as a design variable rather than a fixed constant.

[2] Chen et al. (2014) — timing, energy scale, and GRF magnitude
Chen et al. integrate high-speed kinematics with dynamic analysis, illustrating how elastic energy accumulated in the hind-leg structures is released within a brief time frame. The reported energy scale, on the order of 10 mJ, and the ground reaction forces, reaching multiple body weights, set practical limits on the amount of strain energy that our compliant flexure needs to store and the peak forces that the ground contact and linkage must endure.

[3] Burrows & Sutton (2012) — elastic composite mechanism (stiff + compliant)
This paper explores the mechanisms by which insects store energy, emphasizing that this occurs not through a singular "coil spring," but rather through a composite structure that integrates hard cuticle with highly elastic resilin. This observation directly informs our "no spring/rubber band" constraint: our goal is to more accurately emulate biological systems by incorporating elasticity into the design through compliant folds and flexures, rather than introducing a separate spring element..

[4] Zhang et al. (2019) — rigid–flexible coupled leg linkage lessons
Zhang et al. demonstrate that integrating rigid linkages with compliant elements can enhance jump performance and stability, particularly in uneven terrain. Their findings at the mechanism level reinforce our strategy of employing a foldable four-bar-like structure that incorporates flexibility, which aims to improve energy recovery and decrease sensitivity to surface irregularities..

[5] Eroğlu (2007) — geometric leg ratios and motion sequence for mechanism analogs
Eroğlu presents a biomechanical framework for the translation of biological systems into mechanical mechanisms, focusing on critical factors such as link ratios, angular ranges, and the sequential processes of preload, latch, and extension. This framework serves as a guide for determining optimal link lengths and supports the concept of employing a latch mechanism to effectively separate the dynamics of slow energy storage from the rapid release of that energy.

2.4 Novelty

Previous research typically emphasizes either biological metrics such as takeoff speed and angle or robotic jumping designs that utilize traditional actuation components. The distinctiveness of our project lies in the integration of:
- Foldable robotics manufacturing,
- Structure-embedded compliance (utilizing compliant flexures instead of springs or rubber bands), and
- Multi-surface testing to examine how compliance and latch timing affect jump repeatability and efficiency in real-world scenarios.

Table 1: Biomechanical Parameters of the Grasshopper Hind-leg

Parameter	**Typical Value / Insight**	**Source**
Body mass	$\sim$ 2–3 g (*Schistocerca gregaria*)	Chen et al., 2014
Hind-leg length	Femur $\approx$ 18 mm; Tibia $\approx$ 22 mm	Eroğlu 2007
Take-off speed	3.24 ± 0.03 m s^{-1} (mean measured by motion tracking)	Hawlena et al., 2011 (Fig. 1b)
Take-off angle	$72.6 \pm 0.06°$ from linear fit of trajectory – under threatened conditions	Hawlena et al., 2011 (Fig. 1a)
Take-off angle	40–50° for maximum distance – under unthreatened conditions	Hawlena et al., 2011 (Fig. 1a)
Energy stored per jump	10–14 mJ total in both hind legs (semilunar processes + resilin pads)	Chen 2014; Burrows & Sutton 2012
Ground-reaction force	$\approx$ 8–10$\times$ body weight at take-off; impulse $\approx$ 2–5 ms duration	Chen 2014
Material properties	Composite of **hard chitin** ($E \approx$ 1–3 GPa) + **resilin** ($E \approx$ 1 MPa) $\rightarrow$ high elastic recovery	Burrows & Sutton 2012

Table 2: Leg Motion Sequence and Functional Role of Flexibility

Aspect	**Description**	**Source**
Leg motion sequence	Co-contraction $\rightarrow$ Latch hold $\rightarrow$ Tibia extension $\rightarrow$ Take-off $\rightarrow$ Reset	Eroğlu 2007
Functional role of flexibility	Elastic regions enable energy storage and fast recoil for efficient jumps	Burrows & Sutton 2012

3.Estimated Goal Performance Metrics

In this section we estimate the target performance of our grasshopper-inspired, foldable jumping mechanism. We use biological data for mass and take-off speed to determine the required take-off energy, the equivalent stiffness of the compliant flexure (no discrete spring or rubber band), the torque needed to preload this flexure, and the expected ballistic jump height and range.

1.1 Take-off Energy Requirement

We model the jump as an approximately ballistic take-off with mass m and take-off speed v. The required kinetic energy at the instant of take-off is

$$E_{\text{takeoff}} = \frac{1}{2} m v^2. \tag{1}$$

Using

$$m = 0.0025\,\text{kg}, \qquad v = 3.24\,\text{m/s},$$

we obtain

$$\begin{aligned} E_{\text{takeoff}} &= \frac{1}{2}(0.0025\,\text{kg})\,(3.24\,\text{m/s})^2 \\ &\approx 0.0131\,\text{J} \\ &= 1.31 \times 10^{-2}\,\text{J}. \end{aligned} \tag{2}$$

This is the amount of mechanical energy that must be stored in the compliant flexure and then released at take-off.

1.2 Equivalent Flexure Stiffness (No Discrete Spring)

Instead of using a discrete spring or rubber band, we treat the compliant flexure in the leg as an equivalent linear elastic element with effective stiffness k_{eq} and tip deflection δ. The stored strain energy is

$$E_{\text{takeoff}} \approx \frac{1}{2} k_{\text{eq}} \delta^2, \tag{3}$$

so that

$$k_{\text{eq}} \approx \frac{2E_{\text{takeoff}}}{\delta^2}. \tag{4}$$

For a target deflection

$$\delta = 0.01\,\mathrm{m} \quad (10\,\mathrm{mm}),$$

we get

$$\begin{aligned} k_{\mathrm{eq}} &\approx \frac{2 \times 0.0131\,\mathrm{J}}{(0.01\,\mathrm{m})^2} \\ &\approx 2.62 \times 10^{2}\,\mathrm{N/m}. \end{aligned} \tag{5}$$

The corresponding peak reaction force in the flexure at full preload is

$$\begin{aligned} F_{\mathrm{flex}} &= k_{\mathrm{eq}}\,\delta \\ &\approx (262\,\mathrm{N/m})(0.01\,\mathrm{m}) \\ &\approx 2.62\,\mathrm{N}. \end{aligned} \tag{6}$$

1.3 Torque Required to Preload the Flexure

Let r be the moment arm from the joint axis to the line of action of the flexure force. We take

$$r = 0.015\,\mathrm{m} \quad (15\,\mathrm{mm}),$$

based on the proposed leg geometry. The torque required to preload one leg is

$$\begin{aligned} \tau_{\mathrm{leg}} &= F_{\mathrm{flex}}\,r \\ &\approx (2.62\,\mathrm{N})(0.015\,\mathrm{m}) \\ &\approx 3.9 \times 10^{-2}\,\mathrm{N\,m}. \end{aligned} \tag{7}$$

If a single actuator is used to preload two symmetric hind legs, the total required torque is

$$\begin{aligned} \tau_{\mathrm{total}} &= 2\,\tau_{\mathrm{leg}} \\ &\approx 7.9 \times 10^{-2}\,\mathrm{N\,m}. \end{aligned} \tag{8}$$

This value is used to select an appropriate servo or actuation scheme for slowly cocking the mechanism.

1.4 Ballistic Height and Range

Assuming an ideal ballistic flight with take-off speed v and angle θ, the velocity components are

$$v_x = v\cos\theta, \qquad v_y = v\sin\theta, \tag{9}$$

and the maximum jump height h and horizontal range R (ignoring air drag) are

$$h = \frac{v_y^2}{2g}, \qquad R = \frac{v^2\sin(2\theta)}{g}, \tag{10}$$

where $g = 9.81\,\mathrm{m/s^2}$.

For the biologically observed take-off angle

$$\theta = 72.6^\circ, \qquad v = 3.24\,\mathrm{m/s},$$

we obtain

$$v_x \approx 0.97\,\mathrm{m/s}, \qquad v_y \approx 3.09\,\mathrm{m/s}, \tag{11}$$

$$h \approx 0.49\,\mathrm{m}, \qquad R \approx 0.61\,\mathrm{m}. \tag{12}$$

For comparison, the theoretical maximum range for the same speed occurs at $\theta = 45^\circ$:

$$\begin{aligned} R_{\max} &= \frac{v^2}{g} \\ &= \frac{(3.24\,\mathrm{m/s})^2}{9.81\,\mathrm{m/s^2}} \\ &\approx 1.07\,\mathrm{m}. \end{aligned} \tag{13}$$

These values define the target scale for jump height and distance that our foldable mechanism should aim to approach.

1.5 Average Vertical Acceleration and Ground-Reaction Force

If the vertical take-off velocity v_y is reached over a push-off duration t, the average vertical acceleration during push-off is

$$a_y \approx \frac{v_y}{t}. \tag{14}$$

Using

$$v_y \approx 3.09\,\mathrm{m/s}, \qquad t = 0.05\,\mathrm{s},$$

gives

$$a_y \approx \frac{3.09\,\mathrm{m/s}}{0.05\,\mathrm{s}} \approx 6.18 \times 10^{1}\,\mathrm{m/s^2}. \tag{15}$$

The corresponding average vertical ground-reaction force during push-off is

$$F_{\mathrm{GRF}} \approx m(a_y + g) \approx 0.0025\,\mathrm{kg}\,(61.8 + 9.81)\,\mathrm{m/s^2} \approx 1.8 \times 10^{-1}\,\mathrm{N}. \tag{16}$$

This force level is consistent with literature reports of ground-reaction forces on the order of several times body weight for biological jumpers and provides a design target for the leg structure and foot–ground interface.

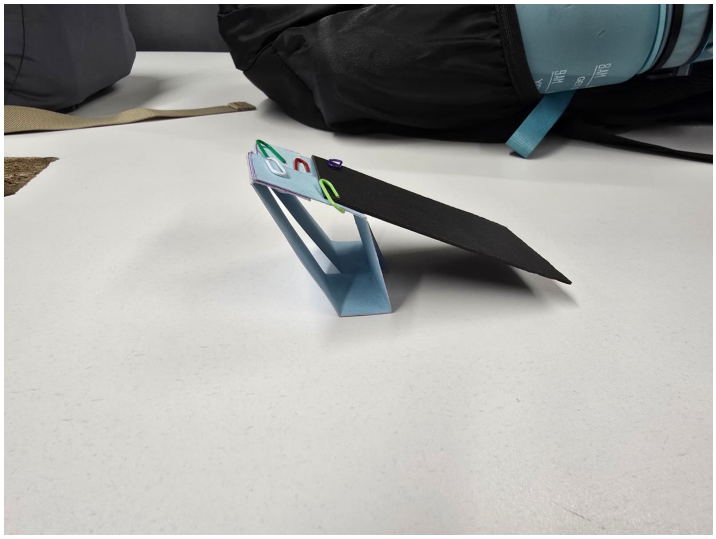
Specification Table

Summarizes the main target specifications and estimated parameters for the grasshopper-inspired jumping mechanism.

Table 1: Summary of target specifications and estimated parameters.

Quantity	Symbol	Value (SI units)
Body mass	m	$0.0025\,\mathrm{kg}$
Take-off speed	v	$3.24\,\mathrm{m/s}$
Take-off angle (escape)	θ	72.6°
Reference max-range angle	$\theta_{\max R}$	45°
Push-off duration	t	$0.05\,\mathrm{s}$
Flexure tip deflection	δ	$0.01\,\mathrm{m}$
Equivalent flexure stiffness	k_{eq}	$2.62 \times 10^{2}\,\mathrm{N/m}$
Stored elastic energy	E_{takeoff}	$1.31 \times 10^{-2}\,\mathrm{J}$
Peak flexure reaction force	F_{flex}	$2.62\,\mathrm{N}$
Moment arm to flexure	r	$0.015\,\mathrm{m}$
Torque per leg to preload	τ_{leg}	$3.9 \times 10^{-2}\,\mathrm{N\,m}$
Total torque (two legs)	τ_{total}	$7.9 \times 10^{-2}\,\mathrm{N\,m}$
Average vertical acceleration	a_y	$6.18 \times 10^{1}\,\mathrm{m/s^2}$
Average vertical GRF	F_{GRF}	$1.8 \times 10^{-1}\,\mathrm{N}$
Gravity	g	$9.81\,\mathrm{m/s^2}$

5. Four bar linkage Mechanism: Prototype



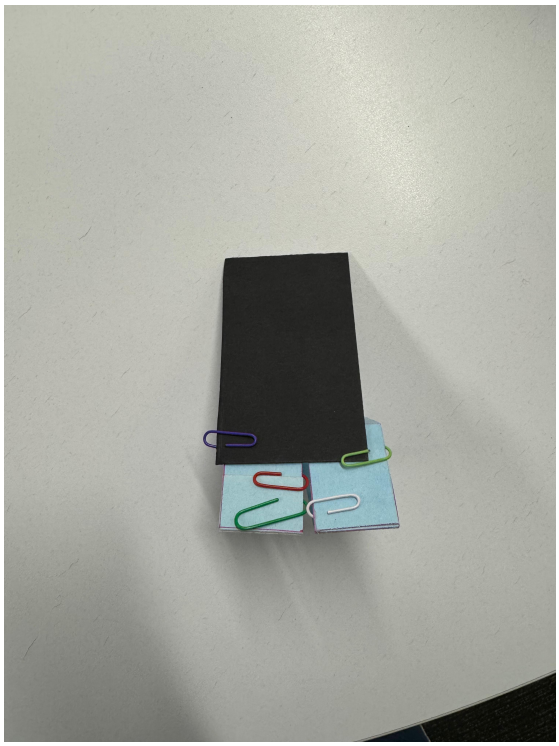


Figure: Prototype for our model

6. Kinematic Model & Jacobian (Planar Linkage): I will attach the Jupyter notebook after that report

7. Modeling and Analysis (Forces, Speeds, Power)

7.1 Why a Latch + Flexure (Instead of High-Speed Motors)

Grasshoppers achieve rapid leg extension by quickly releasing stored elastic energy, while their muscles preload slowly. Our mechanism operates on the same principle: the servo performs a slow cocking motion, and the swift movement results from the rapid elastic recoil upon latch release. This approach is essential, as low-cost micro-servos cannot attain the angular velocities required for a leg extension event lasting 10 to 15 milliseconds.

7.2 End-Effector Force Estimate (Order-of-Magnitude)

A simple estimate uses takeoff time and vertical takeoff velocity. If v_y is achieved over time t, average vertical acceleration is ay ≈vy /t. Then: F≈m(a_y +g),Using the target mass and timing gives forces on the order of 0.14–0.18N, consistent with multiple bodyweights at takeoff.

7.3 Speed Estimate

From the literature timing, a large angular change can occur in ~15 ms, implying ω on the order of 10^2 rad/s. The resulting tip speed v_tip ≈ωL is consistent with the COM takeoff speed order of magnitude. (Again: the motor is not expected to directly produce this speed; latch release is.)

8. Discussion

8.1 Degrees of Freedom and Motors

The planar linkage of each leg can be conceptualized as a four-bar equivalent, which features multiple joint angles yet exhibits fewer independent degrees of freedom (DOF) due to applied constraints. For practical actuation, we consider the system to possess:

- One primary actuated DOF for cocking (representing the preload angle/position), which is mechanically coupled across both legs through a shared shaft, and
- - A binary latch state (either locked or unlocked) that governs the timing of the release.

This design approach aligns with the biological distinction between "slow energy storage" and "fast release," effectively reducing the number of actuators needed while maintaining symmetry in the mechanism.

8.2 How We Estimated End-Effector Forces

In our study, we employed literature-reported peak ground reaction forces (GRF) expressed in multiple bodyweights to facilitate our analysis. To obtain a simplified estimate of impulse and acceleration, we focused on the takeoff time and the velocity components at the moment of takeoff. This approach yields an average force estimate that can then be translated into joint torques through the application of moment arms and Jacobian transpose relationships within the model framework. By integrating these elements, we aim to provide a comprehensive understanding of the biomechanical dynamics at play.

8.3 How We Estimated End-Effector Speeds

We analyzed the takeoff velocity by breaking it down into its horizontal (v_x) and vertical (v_y) components, which were determined based on the angle of takeoff. Additionally, we estimated the foot-tip speed by examining the reported angular changes during the brief extension period. The main takeaway is that the high instantaneous speed is generated by elastic recoil (flexure release) rather than by the output speed of the servos.

10. References (IEEE Style)

[1] D. Hawlena, H. Kress, E. R. Dufresne, and O. J. Schmitz, “Grasshoppers alter jumping biomechanics to enhance escape performance under chronic risk of spider predation,” *Functional Ecology*, vol. 25, pp. 279–288, 2011, doi: 10.1111/j.1365-2435.2010.01767.x.
[2] D. S. Chen, J. M. Yin, K. W. Chen, et al., “Biomechanical and dynamic mechanism of locust take-off,” *Acta Mechanica Sinica*, vol. 30, pp. 762–774, 2014, doi: 10.1007/s10409-014-0065-2.
[3] M. Burrows and G. P. Sutton, “Locusts use a composite of resilin and hard cuticle as an energy store for jumping and kicking,” *J. Exp. Biol.*, vol. 215, no. 19, pp. 3501–3512, 2012, doi: 10.1242/jeb.071993.
[4] A. K. Eroğlu, “Development and Analysis of Grasshopper-Like Jumping Mechanism in Biomimetic Approach,” Ph.D. dissertation, 2007.
[5] Z. Zhang, Q. Yang, J. Zhao, et al., “Dynamic model and performance analysis of rigid-flexible coupling four-bar leg mechanism for small scale bio-inspired jumping robot,” *Microsystem Technologies*, vol. 25, pp. 3269–3285, 2019, doi: 10.1007/s00542-019-04546-5.

Untitled

December 7, 2025

0.1 Kinematic Model

Jacobian Calculations 4 Bar JupyterLab

[3]:
```
import sympy
from sympy import sin, cos
from math import pi
import numpy
import matplotlib.pyplot as plt
import scipy.optimize as so
```

[4]:
```
q1,q2,q3,q4=sympy.symbols('\\theta_1,\\theta_2,\\theta_3,\\theta4')
```

[5]:
```
q1
```

[5]: θ_1

[6]:
```
q2
```

[6]: θ_2

[7]:
```
q3
```

[7]: θ_3

[8]:
```
l0,l1,l2,l3=sympy.symbols('l_0,l_1,l_2,l_3')
```

[9]:
```
l1
```

[9]: l_1

[10]:
```
l2
```

[10]: l_2

[11]:
```
l3
```

[11]: l_3

[12]:
```
def Ry(theta):
    return sympy.
↪Matrix([[cos(theta),0,sin(theta)],[0,1,0],[-sin(theta),0,cos(theta)]])
```

[13]:
```
Ra=Ry(q1)
Rb=Ry(q2)
Rc=Ry(q3)
Rd=Ry(q4)
```

[14]:
```
Ra
```

[14]:
$$\begin{bmatrix} \cos(\theta_1) & 0 & \sin(\theta_1) \\ 0 & 1 & 0 \\ -\sin(\theta_1) & 0 & \cos(\theta_1) \end{bmatrix}$$

[15]:
```
Rb
```

[15]:
$$\begin{bmatrix} \cos(\theta_2) & 0 & \sin(\theta_2) \\ 0 & 1 & 0 \\ -\sin(\theta_2) & 0 & \cos(\theta_2) \end{bmatrix}$$

[16]:
```
Rc
```

[16]:
$$\begin{bmatrix} \cos(\theta_3) & 0 & \sin(\theta_3) \\ 0 & 1 & 0 \\ -\sin(\theta_3) & 0 & \cos(\theta_3) \end{bmatrix}$$

[17]:
```
Rd
```

[17]:
$$\begin{bmatrix} \cos(\theta 4) & 0 & \sin(\theta 4) \\ 0 & 1 & 0 \\ -\sin(\theta 4) & 0 & \cos(\theta 4) \end{bmatrix}$$

[18]:
```
x=sympy.Matrix([1,0,0])
```

[19]:
```
v0_in_n=l0*x
v0_in_n
```

[19]:
$$\begin{bmatrix} l_0 \\ 0 \\ 0 \end{bmatrix}$$

[20]:
```
v1_in_a=l1*x
v1_in_n=Ra*v1_in_a
v1_in_n
```

[20]:
$$\begin{bmatrix} l_1 \cos(\theta_1) \\ 0 \\ -l_1 \sin(\theta_1) \end{bmatrix}$$

[21]:
```
v2_in_b=l2*x
v2_in_a=Rb*v2_in_b
v2_in_n=Ra*v2_in_a
v2_in_n
```

[21]:

$$\begin{bmatrix} -l_2 \sin(\theta_1)\sin(\theta_2) + l_2 \cos(\theta_1)\cos(\theta_2) \\ 0 \\ -l_2 \sin(\theta_1)\cos(\theta_2) - l_2 \sin(\theta_2)\cos(\theta_1) \end{bmatrix}$$

[22]:
```
v3_in_c=l3*x
v3_in_b=Rc*v3_in_c
v3_in_a=Rb*v3_in_b
v3_in_n=Ra*v3_in_a
v3_in_n
```

[22]:
$$\begin{bmatrix} (-l_3 \sin(\theta_2)\sin(\theta_3) + l_3 \cos(\theta_2)\cos(\theta_3))\cos(\theta_1) + (-l_3 \sin(\theta_2)\cos(\theta_3) - l_3 \sin(\theta_3)\cos(\theta_2))\sin(\theta_1) \\ 0 \\ -(-l_3 \sin(\theta_2)\sin(\theta_3) + l_3 \cos(\theta_2)\cos(\theta_3))\sin(\theta_1) + (-l_3 \sin(\theta_2)\cos(\theta_3) - l_3 \sin(\theta_3)\cos(\theta_2))\cos(\theta_1) \end{bmatrix}$$

[23]:
```
p_end=v0_in_n+v1_in_n+v2_in_n+v3_in_n
```

[24]:
```
#Guess the link
design={}
design[l0]= 44.5
design[l1]=33.3
design[l2]=54
design[l3]=13
design
```

[24]:
```
{l_0: 44.5, l_1: 33.3, l_2: 54, l_3: 13}
```

[25]:
```
zero_vec=p_end
zero_vec.simplify()
zero_vec[0]
```

[25]:
$$l_0 + l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2) + l_3 \cos(\theta_1 + \theta_2 + \theta_3)$$

[26]:
```
zero=[]
zero.append((zero_vec.T*sympy.Matrix([1,0,0]))[0])
zero.append((zero_vec.T*sympy.Matrix([0,1,0]))[0])
zero=sympy.Matrix(zero)
zero.simplify()
```

[27]:
```
zero[0]
```

[27]:
$$l_0 + l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2) + l_3 \cos(\theta_1 + \theta_2 + \theta_3)$$

[28]:
```
zero_design = zero.subs(design)
zero_design = zero_design
zero_design
```

[28]:
$$\begin{bmatrix} 33.3 \cos(\theta_1) + 54 \cos(\theta_1 + \theta_2) + 13 \cos(\theta_1 + \theta_2 + \theta_3) + 44.5 \\ 0 \end{bmatrix}$$

[29]:
```
def objective_function(qn):
    q1n,q2n,q3n,q4n=qn
    subs = {}
    subs[q1]=q1n
    subs[q2]=q2n
    subs[q3]=q3n
    subs[q4]=q4n

    zero_n = zero_design.subs(subs)
    sos = ((zero_n.T*zero_n)[0])**.5
    return float(sos)
```

[30]:
```
guess = numpy.array([-76, -126, -60, -8,])*pi/180
guess
```

[30]:
```
array([-1.32645023, -2.19911486, -1.04719755, -0.13962634])
```

[31]:
```
objective_function(guess)
```

[31]:
```
0.6788206643815615
```

[32]:
```
origin = sympy.Matrix([0,0,0])
p0 = v0_in_n
p1 = v0_in_n + v1_in_n
p2 = v0_in_n+v1_in_n + v2_in_n
p3 = v0_in_n+v1_in_n + v2_in_n + v3_in_n
```

[33]:
```
points = sympy.Matrix([p3.T,p2.T,p1.T,p0.T,origin.T]).T
```

[34]:
```
points
```

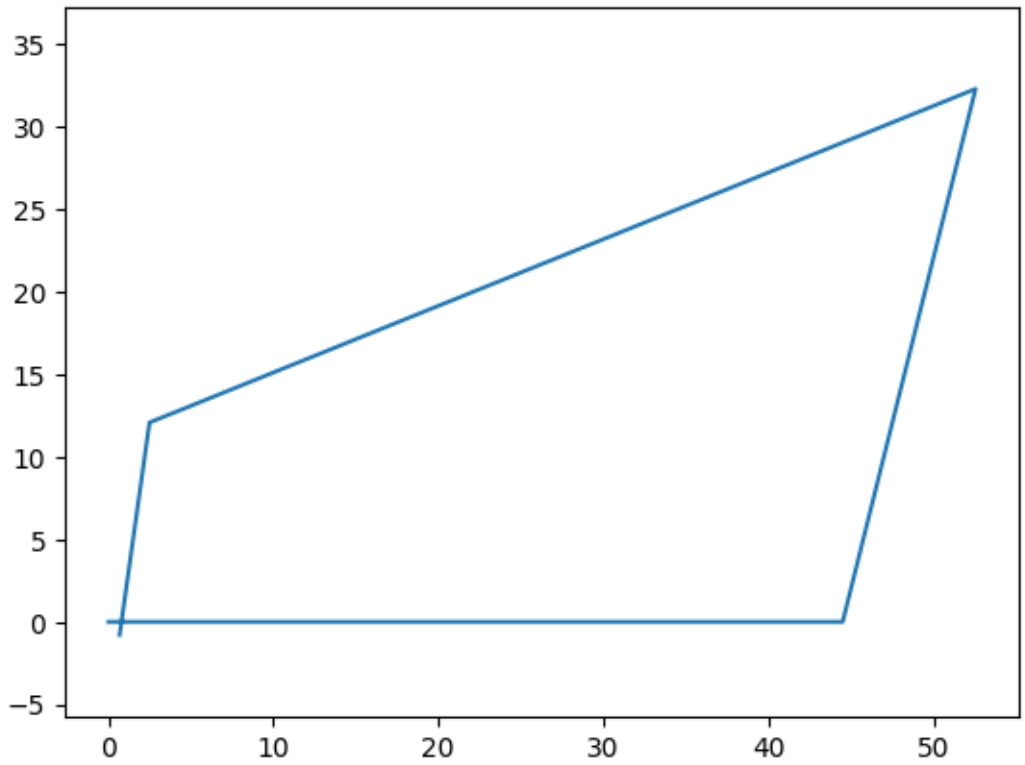
[34]:
$$\begin{bmatrix} l_0 + l_1 \cos(\theta_1) - l_2 \sin(\theta_1)\sin(\theta_2) + l_2 \cos(\theta_1)\cos(\theta_2) + (-l_3 \sin(\theta_2)\sin(\theta_3) + l_3 \cos(\theta_2)\cos(\theta_3))\cos(\theta_1) + (-l_3 s \\ 0 \\ -l_1 \sin(\theta_1) - l_2 \sin(\theta_1)\cos(\theta_2) - l_2 \sin(\theta_2)\cos(\theta_1) - (-l_3 \sin(\theta_2)\sin(\theta_3) + l_3 \cos(\theta_2)\cos(\theta_3))\sin(\theta_1) + (-l_3 \sin \end{bmatrix}$$

[35]:
```
points_design = points.subs(design)
```

[36]:
```
def plot_fourbar(thetas):
    state_variables = q1,q2,q3,q4
    state = dict(zip(state_variables,thetas))
    points_state = points_design.subs(state)
    points_state_numpy = numpy.array(points_state,dtype=numpy.float64)
    print(points_state_numpy)
    plt.plot(points_state_numpy[0,:],points_state_numpy[2,:])
    plt.axis('equal')
```

[37]:
```
plot_fourbar(guess)
```

```
[[ 0.67882066  2.48807098 52.55599912 44.5         0.        ]
 [ 0.          0.          0.          0.          0.        ]
 [-0.79139325 12.08209164 32.31084768  0.          0.        ]]
```



[38]:
```
zero=sympy.Matrix(zero)
zero.simplify()
zero
```

[38]:
$$\begin{bmatrix} l_0 + l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2) + l_3 \cos(\theta_1 + \theta_2 + \theta_3) \\ 0 \end{bmatrix}$$

[39]:
```
independent=sympy.Matrix([q1])
dependent=sympy.Matrix([q1,q2,q3,q4])
zero_design=zero.subs(design)
```

[40]:
```
A=zero_design.jacobian(independent)
```

[41]:
```
A
```

[41]:
$$\begin{bmatrix} -33.3 \sin(\theta_1) - 54 \sin(\theta_1 + \theta_2) - 13 \sin(\theta_1 + \theta_2 + \theta_3) \\ 0 \end{bmatrix}$$

[42]:
```
B=zero_design.jacobian(dependent)
```

[43]:
```
B
```

[43]:
$$\begin{bmatrix} -33.3\sin(\theta_1) - 54\sin(\theta_1+\theta_2) - 13\sin(\theta_1+\theta_2+\theta_3) & -54\sin(\theta_1+\theta_2) - 13\sin(\theta_1+\theta_2+\theta_3) & -13\sin(\theta_1+\theta_2 \\ 0 & 0 & 0 \end{bmatrix}$$

[44]:
```
result1 = so.minimize(objective_function,guess,options={'xrtol':1e-4})
state_variables = q1,q2,q3,q4
state = dict(zip(state_variables,result1.x))
An=numpy.array(A.subs(state),dtype=float)
An
```

[44]:
```
array([[0.18390222],
       [0.        ]])
```

[45]:
```
Bn=numpy.array(B.subs(state),dtype=float)
Bn
```

[45]:
```
array([[  0.18390222, -32.12346704, -12.82318155,   0.        ],
       [  0.        ,   0.        ,   0.        ,   0.        ]])
```

[46]:
```
print(Bn.shape)
```

```
(2, 4)
```

[47]:
```
import numpy as np
Cn,*_=np.linalg.lstsq(Bn,-An,rcond=None)
Cn
```

[47]:
```
array([[-2.82685162e-05],
       [ 4.93785633e-03],
       [ 1.97111439e-03],
       [ 0.00000000e+00]])
```

[48]:
```
p_out=v3_in_n
p_out
```

[48]:
$$\begin{bmatrix} (-l_3\sin(\theta_2)\sin(\theta_3) + l_3\cos(\theta_2)\cos(\theta_3))\cos(\theta_1) + (-l_3\sin(\theta_2)\cos(\theta_3) - l_3\sin(\theta_3)\cos(\theta_2))\sin(\theta_1) \\ 0 \\ -(-l_3\sin(\theta_2)\sin(\theta_3) + l_3\cos(\theta_2)\cos(\theta_3))\sin(\theta_1) + (-l_3\sin(\theta_2)\cos(\theta_3) - l_3\sin(\theta_3)\cos(\theta_2))\cos(\theta_1) \end{bmatrix}$$

[49]:
```
D = p_out.jacobian(independent)
Dn = numpy.array(D.subs(design).subs(state),dtype=float)
E = p_out.jacobian(dependent)
En = numpy.array(E.subs(design).subs(state),dtype=float)
Jo = Dn+(En@Cn)
Jo
```

[49]:
```
array([[-12.91141404],
       [  0.        ],
       [  2.15152643]])
```

[50]:
```
y_dot = Jo@numpy.array([[1,]]).T
y_dot
```

[50]:
```
array([[-12.91141404],
       [  0.        ],
       [  2.15152643]])
```

[51]:
```
p_out_n = numpy.array(p_out.subs(design).subs(state),dtype=float)
p_out_n
```

[51]:
```
array([[ -2.13682358],
       [  0.        ],
       [-12.82318155]])
```

Another code analysis for Jacobian

[53]:
```
import sympy as sp

# ============================================================
# Another code for Jacobian: planar 4-bar linkage
# ============================================================

# 1. Symbols (link lengths and joint angles)
L1, L2, L3, L4 = sp.symbols('L1 L2 L3 L4', positive=True)

# Joint angles (in radians)
# theta2: input rocker
# theta3: coupler
# theta4: output rocker
theta2, theta3, theta4 = sp.symbols('theta2 theta3 theta4', real=True)

# Time derivatives of the joint angles
theta2_dot, theta3_dot, theta4_dot = sp.symbols(
    'theta2_dot theta3_dot theta4_dot', real=True
)

# ------------------------------------------------------------
# 2. Geometry of the 4-bar in the plane
# ------------------------------------------------------------

# Ground pivots: O2 at (0, 0), O4 at (L1, 0)
O2 = sp.Matrix([0, 0])
O4 = sp.Matrix([L1, 0])

# Input rocker (link 2): from O2 to point A
A = O2 + sp.Matrix([
    L2 * sp.cos(theta2),
    L2 * sp.sin(theta2)
```

```
])

# Output rocker (link 4): from O4 to point B
B = O4 + sp.Matrix([
    L4 * sp.cos(theta4),
    L4 * sp.sin(theta4)
])

# Coupler (link 3) from A to B must have length L3
# Loop-closure: A + L3*[cos(theta3), sin(theta3)] = B
closure = B - A - sp.Matrix([
    L3 * sp.cos(theta3),
    L3 * sp.sin(theta3)
])

print("============================================================")
print("Loop-closure equations f(theta2, theta3, theta4) = 0")
print("------------------------------------------------------------")
sp.pprint(closure)

# ------------------------------------------------------------
# 3. Differentiate closure equations w.r.t. time
#     (constraint Jacobian)
# ------------------------------------------------------------

df_dtheta2 = sp.diff(closure, theta2)
df_dtheta3 = sp.diff(closure, theta3)
df_dtheta4 = sp.diff(closure, theta4)

eq_time = (
    df_dtheta2 * theta2_dot
    + df_dtheta3 * theta3_dot
    + df_dtheta4 * theta4_dot
)

print("\n============================================================")
print("Time-differentiated closure equations (must equal zero)")
print("------------------------------------------------------------")
sp.pprint(eq_time)

# We require eq_time[0] = 0 and eq_time[1] = 0

# ------------------------------------------------------------
# 4. Solve the 2x2 linear system for theta3_dot and theta4_dot
# ------------------------------------------------------------

solution = sp.solve(
```

```
    [eq_time[0], eq_time[1]],
    [theta3_dot, theta4_dot],
    dict=True
)[0]

theta3_dot_expr = sp.simplify(solution[theta3_dot])
theta4_dot_expr = sp.simplify(solution[theta4_dot])

print("\n============================================================")
print("Angular rates of other links expressed in terms of theta2_dot")
print("------------------------------------------------------------")
print("theta3_dot(theta2) =")
sp.pprint(theta3_dot_expr)
print("\ntheta4_dot(theta2) =")
sp.pprint(theta4_dot_expr)

# Partial derivatives dtheta3/dtheta2 and dtheta4/dtheta2
dtheta3_dtheta2 = sp.simplify(theta3_dot_expr / theta2_dot)
dtheta4_dtheta2 = sp.simplify(theta4_dot_expr / theta2_dot)

print("\n============================================================")
print("Partial derivatives dtheta3/dtheta2 and dtheta4/dtheta2")
print("------------------------------------------------------------")
print("dtheta3_dtheta2 =")
sp.pprint(dtheta3_dtheta2)
print("\ndtheta4_dtheta2 =")
sp.pprint(dtheta4_dtheta2)

# ------------------------------------------------------------
# 5. End-effector position and Jacobian
#    Here: end-effector = point B (end of link 4)
# ------------------------------------------------------------

x_B, y_B = B[0], B[1]

# Chain rule: d/dtheta2 of x_B and y_B,
# accounting for theta3(theta2) and theta4(theta2)
Jx = sp.simplify(
    sp.diff(x_B, theta2)
    + sp.diff(x_B, theta3) * dtheta3_dtheta2
    + sp.diff(x_B, theta4) * dtheta4_dtheta2
)
Jy = sp.simplify(
    sp.diff(y_B, theta2)
    + sp.diff(y_B, theta3) * dtheta3_dtheta2
    + sp.diff(y_B, theta4) * dtheta4_dtheta2
)
```

```python
print("\n============================================================")
print("Jacobian mapping theta2_dot to Cartesian velocity of point B")
print("------------------------------------------------------------")
print("dx_B/dtheta2 =")
sp.pprint(Jx)
print("\ndy_B/dtheta2 =")
sp.pprint(Jy)

# If you want actual velocity for a given theta2_dot:
vx_B = sp.simplify(Jx * theta2_dot)
vy_B = sp.simplify(Jy * theta2_dot)

print("\nVelocity of B for a given theta2_dot:")
print("vx_B =")
sp.pprint(vx_B)
print("\nvy_B =")
sp.pprint(vy_B)
```

```
============================================================
Loop-closure equations f(theta2, theta3, theta4) = 0
------------------------------------------------------------
 L  - L cos( ) - L cos( ) + L cos( )

   -L sin( ) - L sin( ) + L sin( )

============================================================
Time-differentiated closure equations (must equal zero)
------------------------------------------------------------
 L  _2_dot sin( ) + L  _3_dot sin( ) - L  _4_dot sin( )

 -L  _2_dot cos( ) - L  _3_dot cos( ) + L  _4_dot cos( )

============================================================
Angular rates of other links expressed in terms of theta2_dot
------------------------------------------------------------
theta3_dot(theta2) =
-L  _2_dot sin(  -  )

     L sin(  -  )

theta4_dot(theta2) =
-L  _2_dot sin(  -  )

     L sin(  -  )

============================================================
Partial derivatives dtheta3/dtheta2 and dtheta4/dtheta2
```

```
------------------------------------------------------------
dtheta3_dtheta2 =
-L sin(  -  )

 L sin(  -  )

dtheta4_dtheta2 =
-L sin(  -  )

 L sin(  -  )

============================================================
Jacobian mapping theta2_dot to Cartesian velocity of point B
------------------------------------------------------------
dx_B/dtheta2 =
L sin( ) sin(  -  )

     sin(  -  )

dy_B/dtheta2 =
-L sin(  -  ) cos( )

      sin(  -  )

Velocity of B for a given theta2_dot:
vx_B =
L  _2_dot sin( ) sin(  -  )

         sin(  -  )

vy_B =
-L  _2_dot sin(  -  ) cos( )

          sin(  -  )
```

[ ]: