Project Assignment 1
RAS 557
Foldable Robotics
Group 8

Vamshi Narayana Babu
Shawn Dimang
Sameerjeet Singh Chhabra

Project Goal

We aim to study and replicate the biomechanics of the grasshopper's hind-leg jumping mechanism — specifically how its flexible exoskeletal structures and elastic energy storage contribute to powerful, efficient jumps across different terrains.

Our goal is to translate these biological principles into an engineered system: a foldable, bio-inspired jumping mechanism that uses flexible materials to store and release energy efficiently.

Research question: How can bio-inspired grasshopper hind leg mechanisms use flexible material to replicate jumping?

Design/Prototype:
A foldable, flexible-material jumping leg inspired by grasshopper biomechanics, which demonstrates high energy efficiency and robustness on multiple surfaces.

Why this matters:
Grasshoppers achieve some of the most efficient power-to-mass jump performances in nature by combining **stiff cuticular structures** with **compliant resilient pads** and **mechanical latches**. Translating this system into a foldable robotic mechanism offers insight into **low-cost, lightweight, and high-power actuation** — especially valuable for **small mobile robots** or **deployable exploration systems** where energy storage and terrain adaptability are key.

Scope
To keep the project feasible within an 8-week semester, we are constraining our study to the **hind-leg jumping mechanism of the grasshopper** and its **translation into a single foldable prototype leg**. Although we will study and model **one hind leg** in detail (to understand its mechanics, forces, and energy storage), the **prototype will feature two synchronized hind legs** to better replicate **grasshopper-like jumping dynamics**. Instead of attempting to replicate full-body dynamics or flight trajectories, our focus is on **energy storage, release, and takeoff efficiency**. The system will be modeled as a **foldable linkage with a compliant flexure acting as the energy storage element**, using accessible materials (YET TO BE DECIDED). Analytical and simulation-based modeling will guide parameter selection (spring stiffness, leg angle, and latch geometry), followed by **one physical prototype** for experimental validation of jump height, distance, and efficiency. This constrained scope allows us to deliver measurable results—quantitative tests of how flexible materials influence jump performance—while staying within realistic fabrication, testing, and reporting timelines.

Impact

This project is both **scientifically interesting and practically relevant**. Nature's jumping mechanisms are highly optimized for power density and efficiency—understanding and replicating them can inspire more energy-efficient and lightweight robotic systems. Today, there is growing interest in **micro-robots and field-deployable systems** (for inspection, search-and-rescue, and space exploration) that need efficient motion in variable terrain. Advances in **origami-inspired design, flexible composites, and low-cost laser cutting** now make it possible to model and build small-scale bio-inspired mechanisms accurately. The broader impact of this work lies in improving **terrain-adaptive, power-efficient locomotion**. By demonstrating that foldable, flexible designs can reproduce the performance of natural jumpers, the project can inform new directions in **soft robotics, deployable mechanisms, and sustainable material use** for robotics education and research.

Team Fit

Answering this question aligns closely with our team's interests in **biomechanics, energy-efficient robotics, and material-based actuation**. It combines our background knowledge in **mechanical design and dynamics** with our curiosity about **how biological systems achieve high performance through structure rather than complex control**. The project lets us apply skills in **modeling (linkage kinematics, spring energy analysis), rapid prototyping (foldable and flexible fabrication), and experimental testing (high-speed motion capture, efficiency measurement)**. Our interest in using minimal actuation and maximizing mechanical intelligence matches the essence of grasshopper jumping—where morphology and material properties, not electronics, drive performance. By focusing on this topic, we can deepen our understanding of how **mechanical design and material selection together enable efficient locomotion**, which directly supports future work in foldable robotics or bioinspired design research.

Topic Fit

Our research question directly leverages **foldable robotics principles** as both a modeling and fabrication framework. The grasshopper leg will be represented as a **foldable linkage system**—a simplified mechanical analog of the femur-tibia assembly—with **compliant joints and elastic folds** serving as the energy storage and release elements. Using **origami-inspired design techniques**, we can approximate complex 3D motions with planar folds that are easy to fabricate, lightweight, and capable of storing strain energy within their material layers. Foldable mechanisms also make it possible to **integrate stiffness variation and latching behavior** without bulky hardware—mirroring the biological combination of hard cuticle and soft resilin in the grasshopper leg. In this way, foldable robotics is not only a fabrication method but also a **conceptual tool for exploring how geometry and material flexibility interact to produce efficient motion**.

Background

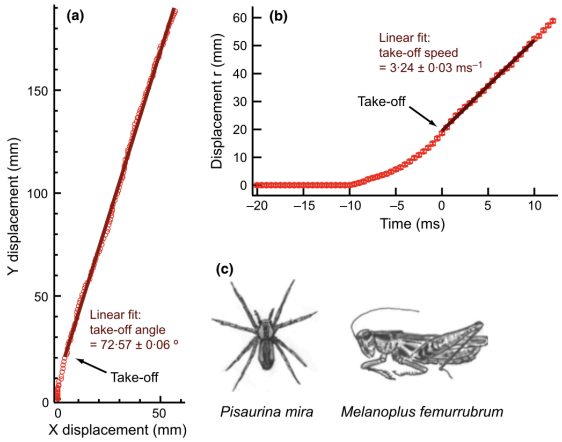


Figure 1(Hawlena et al., 2011)

(a) Plots the **vertical (y)** vs. **horizontal (x)** displacement of the grasshopper's center of mass during take-off. The linear fit shows a **take-off angle of ≈ 72.6 ± 0.06°**, representing the direction of launch.
(b) Shows total displacement *r(t)* vs. time, with a **linear fit for take-off speed ≈ 3.24 ± 0.03 m s^{-1}**.
(c) Illustrates the study organisms – the grasshopper (*Melanoplus femurrubrum*) and its predator (*Pisaurina mira*).

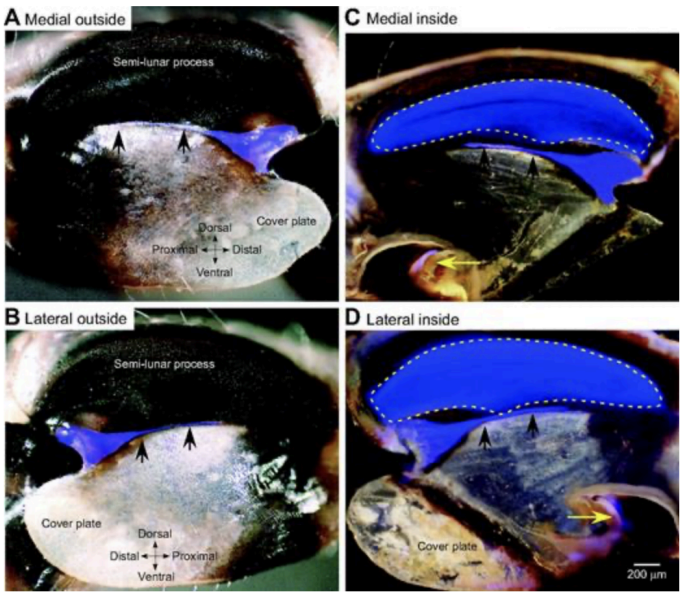
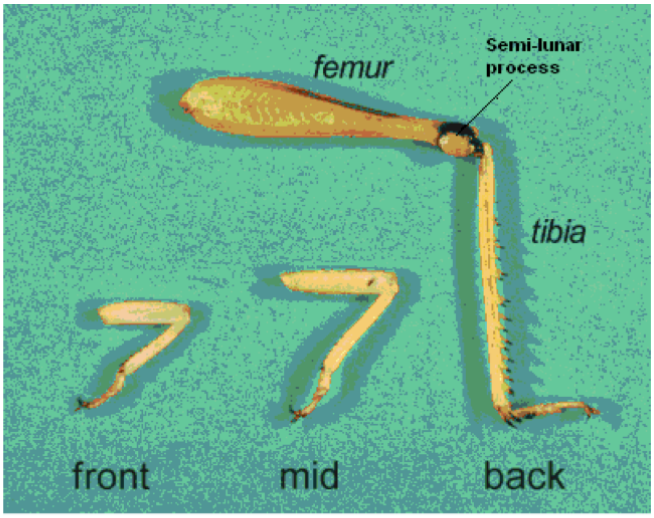


Figure 2 (Burrows & Sutton 2012**)**

Panels A–D show **confocal micrographs** of the locust's hind femur, focusing on the **semi-lunar process** — the key elastic storage structure responsible for powering the jump.

- **(A, B)** show external (medial and lateral) views with the semi-lunar process highlighted.
- **(C, D)** show internal (medial and lateral) cross-sections, where the **blue-shaded area** marks regions rich in **resilin**, a highly elastic protein embedded within the hard chitinous cuticle.
- The yellow arrows indicate the direction of deformation during energy storage.



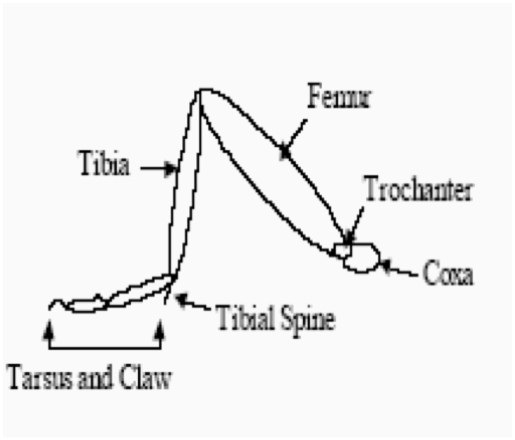
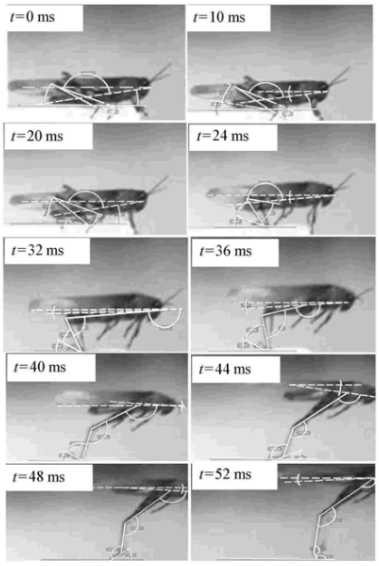


Figure 2.2 Hind leg with segments identified (Fauske, 2002; Laksanacharoen, Pollack, et al., 2005; Laksanacharoen, Quinn and Ritzmann, 2003).



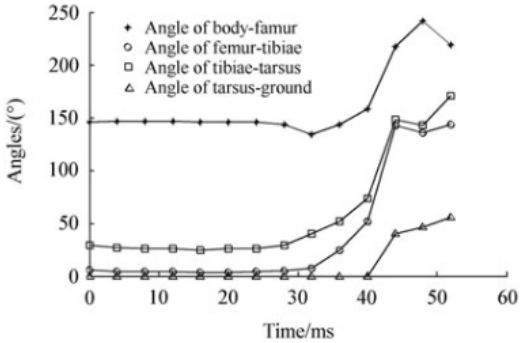


Figure 3 (Chen et al. (2014)

(a) A high-speed camera (500 fps) captures the **take-off sequence** of a locust over 0–52 ms. The frames show the progression from the **pre-loading phase** (t = 0 ms) through **leg extension** (≈ 40 ms) to **airborne take-off** (~ 52 ms).
(b) Plots the **angles** of the body–femur, femur–tibia, tibia–tarsus, and tarsus–ground as functions of time (0–60 ms).

Discussion – Value of Each Paper

1)Grasshoppers alter jumping biomechanics to enhance escape performance under chronic risk of spider predation - Dror Hawlena

This paper provides precise **take-off kinematics** for grasshoppers using high-speed imaging. The authors measured a **take-off speed of 3.24 m/s** and **angle of 72.6°**, which define realistic performance targets for bio-inspired jumpers. Their findings show how grasshoppers adjust joint angles and pre-loading based on environmental pressure, demonstrating the importance of **adaptive biomechanics**. For our project, these data guide the **target motion and efficiency benchmarks** for the foldable leg design, ensuring the prototype mimics natural speed and trajectory.

2) Biomechanical and dynamic mechanism of locust take-off - Chen et al., 2014

Chen et al. (2014) present a dynamic model of **locust take-off mechanics**, combining kinematic data and force analysis. Their **high-speed footage (0–52 ms)** and joint-angle plots reveal how elastic energy stored in the **semi-lunar process** converts to rapid extension and lift. The study quantifies **energy output (~10–14 mJ)** and **ground-reaction forces (8–10× body weight)**, giving mechanical parameters for energy release and spring stiffness.

3) Dynamic model and performance analysis of a rigid–flexible coupling four-bar leg for a small bio-inspired jumping robot. - Zhang, J., Chen, W., Wang, J., & Liu, Z. (2019)

Zhang et al. developed a **rigid–flexible coupled four-bar leg** for a small jumping robot, showing that incorporating compliant elements improves jump stability and distance. Their model demonstrates how flexibility can increase **energy storage and recovery**, especially on uneven terrain. This research directly parallels our project's intent to use **foldable, flexible materials** to improve energy efficiency and multi-terrain performance.

4) Design of a Grasshopper-like Jumping Mechanism in a Biomimetic Approach. - Eroğlu, A. K. (2007)

Eroğlu's thesis translates **grasshopper anatomy** into a functional mechanical model, detailing **femur–tibia geometry, link ratios, and spring mechanics**. It provides the foundational parameters—leg lengths, angular ranges, and material constraints—that make biological motion achievable in a robotic form. For our project, this serves as the **geometric and conceptual baseline** for designing a foldable leg that reproduces the natural motion sequence while incorporating flexible components for energy efficiency.

5) Locusts use a composite of resilin and hard cuticle as an energy store for jumping - Burrows, M., & Sutton, G. P. (2012)
Explains the material science of energy storage in insects. By quantifying stiffness differences between resilin and cuticle, it guides our **flexible material selection** for the foldable mechanism.

Novelty of Our Project

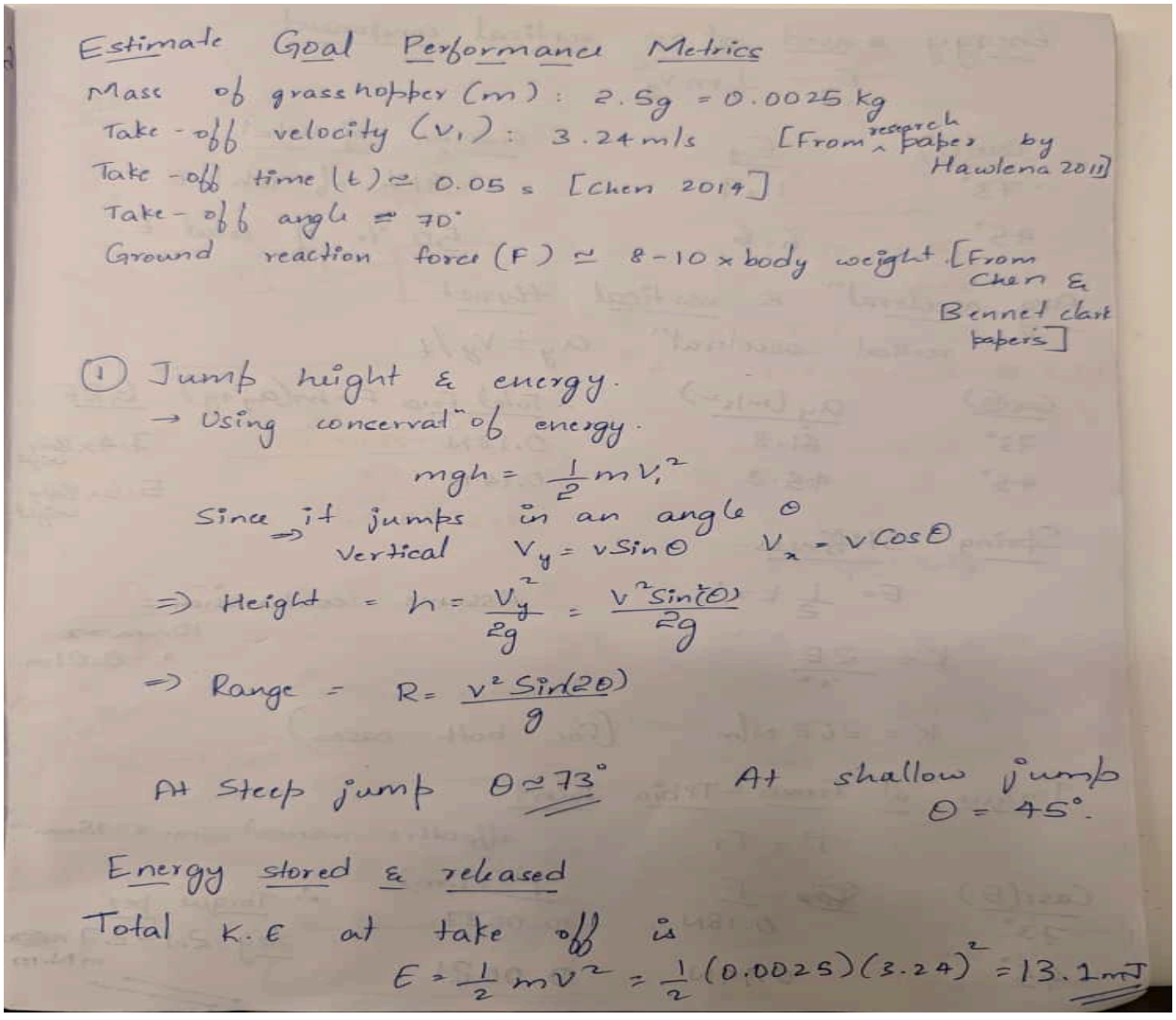
Previous research has focused separately on either the **biomechanics** (Hawlena, Chen, Burrows) or **robotic translation**(Zhang, Eroğlu) of the grasshopper jump. Our project uniquely combines both through **foldable robotics techniques** and **flexible materials** to replicate the biological stiffness–compliance composite within a single mechanism. By modeling and prototyping a **foldable two-leg jumping system** that adapts to different terrains, we aim to demonstrate improved jump efficiency and reusability. This integration of **biological material principles, origami-inspired fabrication, and multi-terrain testing** defines the novelty and relevance of our research within bio-inspired robotics.

Parameter	Typical Value / Insight	Source
Body mass	~2–3 g (*Schistocerca gregaria*)	Chen et al., 2014
Hind-leg length	Femur ≈ 18 mm Tibia ≈ 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 ± 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 (GRF)	≈ 8–10 × body weight at take-off; impulse ≈ 2–5 ms duration	Chen 2014
Material properties of hind-leg cuticle	Composite of **hard chitin (E ≈ 1–3 GPa)** + **resilin (E ≈ 1 MPa)**→ high elastic recovery	Burrows & Sutton 2012

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

Estimated Goal Performance Metrics

To define target performance for our foldable, grasshopper-inspired jumping mechanism, we estimate the **velocity, height, spring energy, and torque** required for efficient take-off based on biological data and mechanical analogs.



Energy used on vertical component

$E_y = \frac{1}{2} m v_y^2$

Case(θ)	E_y	Fraction of total
73°	11.9	91% of total E
45°	6.6	50 % of total E.

Avg acceleratⁿ & vertical thrust

vertical acceleratⁿ $a_y = V_y / t$

Case(θ)	a_y (m/s²)	∴ Total Force $F = m(a_y + g)$	GRF
73°	61.8	0.18 N	7.4 × Body weight
45°	45.8	0.14 N	5.6 × Body weight

Spring Stiffness

$E = \frac{1}{2} k x^2$, Assume compression = 10mm = x

$x = 0.01m$

$k = \frac{2E}{x^2}$

$K = 262 N/m$ (For both cases)

Torque at Femur – Tibia Joint

$T = Fr$ effective moment arm r = 15mm

Case(θ)	F	T (N.m)	∴ Torque per leg 2.1 – 2.7 mN.m
73°	0.18N	0.0027	
45°	0.14N	0.0021	

→ The torque we found is for instantaneous torque during the push-off. the motor does not supply this (the spring does).

→ Preload torque - the motor supply to wind the spring is given by.

$F_s = kx$ [k = 262N/m, x = 0.01]

$\Rightarrow F_s = 2.62 N$

$\therefore T = F_s r$ [r = 0.015m]

$\Rightarrow T = 0.039 N.m$ per leg

∴ for both leg we require total $T \simeq 0.078 N.m$

→ This is the T required by servo motor

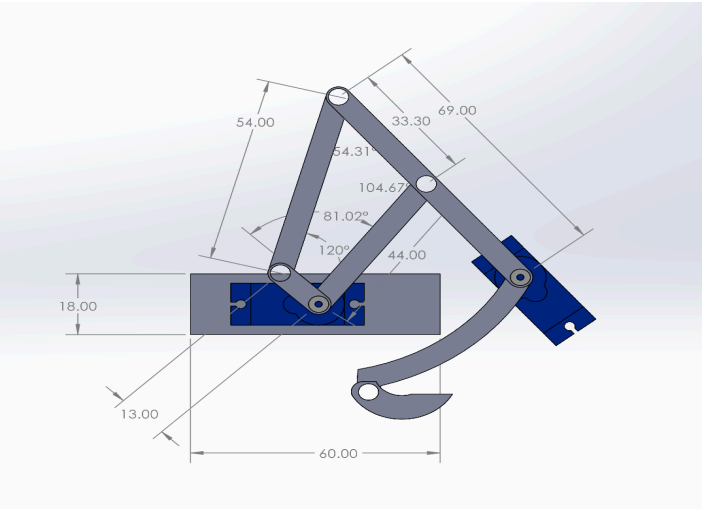
Specifications Table

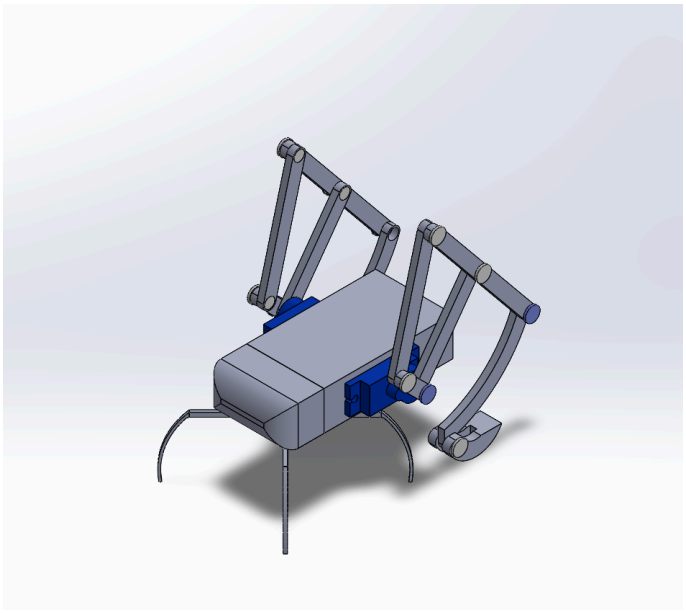
Parameter	Symbol	Value	Units	Description / Source
Mass of system (robot or equivalent)	(m)	0.0025	kg	Approx. grasshopper body mass (Hawlena et al., 2011; Chen et al., 2014)
Take-off velocity	(v)	3.24	m/s	Measured mean from *Hawlena et al.* (2011)
Take-off angle (biological case)	(theta1)	72.6	°	From trajectory fit in *Hawlena et al.*
Take-off angle (optimized for range)	(theta2)	45	°	Theoretically optimal ballistic angle
Take-off time	(t)	0.05	s	Time of leg extension (Chen et al., 2014)
Gravity	(g)	9.81	m/s^2	Standard Earth gravity
Spring compression distance	(x)	0.01	m	Estimated maximum flexure deflection (10 mm)

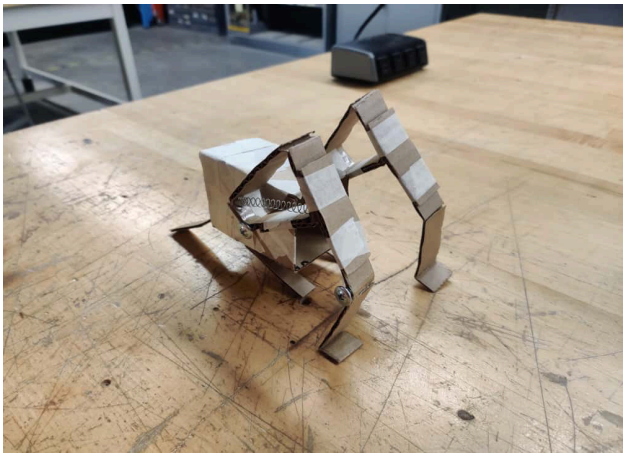
Effective spring stiffness	(k)	262	N/m	From (k = 2E / x^2) using (E = 13.1) mJ
Elastic energy stored	(E)	0.0131	J (13.1 mJ)	Computed from kinetic energy
Spring force at full preload	(F_s)	2.62	N	(F_s = kx)
Moment arm (distance from joint pivot to spring line)	(r)	0.015	m	Approx. 15 mm from design geometry
Torque per leg (cocking / release)	(T1)	0.039	N·m	(T1 = F_s r) for one leg
Torque for both legs (if single motor)	(T2)	0.078	N·m	(T2 = 2T2)
Ground reaction force (peak)	(F_grf)	0.18	N	≈ 7–8× body weight (Chen et al., 2014)

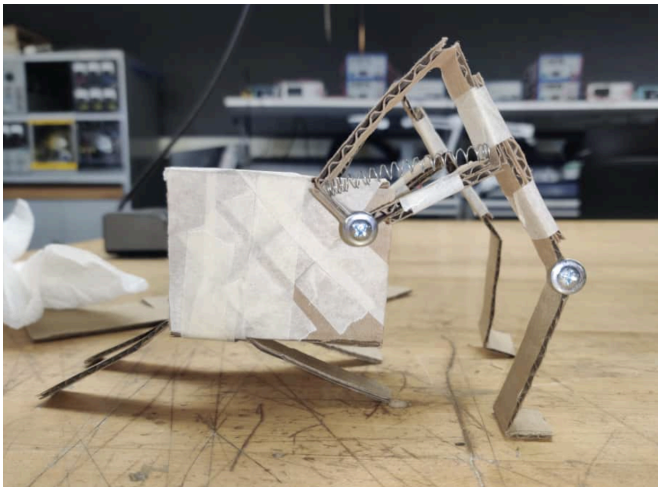
Vertical velocity component (72.6°)	(v_y)	3.09	m/s	
Horizontal velocity component (72.6°)	(v_x)	0.95	m/s	
Jump height (72.6°)	(h)	0.49	m	(h = v_y^2 / 2g)
Jump range (72.6°)	(R)	0.61	m	(R = v^2\sin(2(theta)/g)
Jump range (45°)	(R_max)	1.07	m	Maximum theoretical range

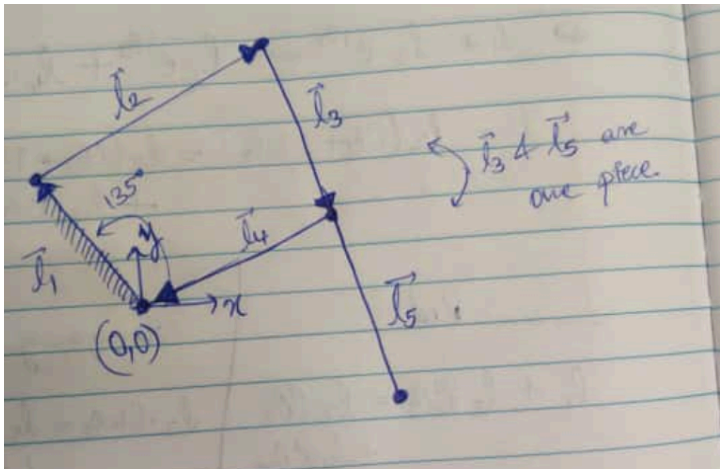
Mechanism: Solidworks and Cardboard Design










$\vec{l}_2$
$\vec{l}_3$
$\vec{l}_3$ & $\vec{l}_5$ are one piece.
135°
$\vec{l}_4$
$\vec{l}_1$
x
$\vec{l}_5$
(0,0)

Kinematic Model

Jacobian Calculations 4 Bar JupyterLab

In [2]:
```
import sympy
from sympy import sin,cos
from math import pi
import numpy
import matplotlib.pyplot as plt
import scipy.optimize as so
```

In [3]:
```
q1,q2,q3,q4 = sympy.symbols('\\theta_1,\\theta_2,\\theta_3,\\theta_4')
```

In [4]:
```
q1
```

Out[4]: θ_1

In [5]:
```
l0,l1,l2,l3 = sympy.symbols('l_0,l_1,l_2,l_3')
l1
```

Out[5]: l_1

In [6]:
```
def Ry(theta):
    return sympy.Matrix([[cos(theta),0,sin(theta)],
                         [0,1,0],
                         [-sin(theta),0,cos(theta)]])
```

In [7]:
```
Ra = Ry(q1)
Rb = Ry(q2)
Rc = Ry(q3)
Rd = Ry(q4)
Ra
```

Out[7]:
$$\begin{bmatrix} \cos(\theta_1) & 0 & \sin(\theta_1) \\ 0 & 1 & 0 \\ -\sin(\theta_1) & 0 & \cos(\theta_1) \end{bmatrix}$$

In [8]:
```
x = sympy.Matrix([1,0,0])
```

In [9]:
```
v0_in_n=l0*x
#v1_in_n=R
v0_in_n
```

Out[9]:
$$\begin{bmatrix} l_0 \\ 0 \\ 0 \end{bmatrix}$$

In [10]:
```
v1_in_a = l1*x
v1_in_n = Ra*v1_in_a
v1_in_n
```

Out[10]:
$$\begin{bmatrix} l_1 \cos(\theta_1) \\ 0 \\ -l_1 \sin(\theta_1) \end{bmatrix}$$

In [11]:
```
v2_in_b = l2*x
v2_in_a = Rb*v2_in_b
v2_in_n = Ra*v2_in_a
v2_in_n
```

Out[11]:
$$\begin{bmatrix} -l_2 \sin(\theta_1)\sin(\theta_2) + l_2 \cos(\theta_1)\cos(\theta_2) \\ 0 \\ -l_2 \sin(\theta_1)\cos(\theta_2) - l_2 \sin(\theta_2)\cos(\theta_1) \end{bmatrix}$$

In [12]:
```
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
```

Out[12]:
$$\begin{bmatrix} (-l_3 \sin(\theta_2)\sin(\theta_3) + l_3 \cos(\theta_2)\cos(\theta_3))\cos(\theta_1) + (-l_3 \sin(\theta_2)\cos(\theta_3) - l_3 \sin(\theta_3)\cos(\theta_2)) \\ 0 \\ -(-l_3 \sin(\theta_2)\sin(\theta_3) + l_3 \cos(\theta_2)\cos(\theta_3))\sin(\theta_1) + (-l_3 \sin(\theta_2)\cos(\theta_3) - l_3 \sin(\theta_3)\cos(\theta_2)) \end{bmatrix}$$

In [13]:
```
#v1_in_n.subs({q1:pi*30/180,l1:1})
```

In [14]:
```
p_end = v0_in_n + v1_in_n + v2_in_n + v3_in_n
```

In [15]:
```
# LINK GUESS / DESIGN
design = {}
design[l0] = 44
design[l1] = 33.3
design[l2] = 54
design[l3] = 13

design
```

Out[15]:
```
{l_0: 44, l_1: 33.3, l_2: 54, l_3: 13}
```

In [16]:
```
zero_vec = p_end
zero_vec.simplify()
zero_vec[0]
```

Out[16]:
$$l_0 + l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2) + l_3 \cos(\theta_1 + \theta_2 + \theta_3)$$

In [17]:
```
zero = []
zero.append((zero_vec.T*sympy.Matrix([1,0,0]))[0])
zero.append((zero_vec.T*sympy.Matrix([0,1,0]))[0])
```

In [18]:
```
zero = sympy.Matrix(zero)
zero.simplify()
```

```
zero[0]
```

Out[18]: $l_0 + l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2) + l_3 \cos(\theta_1 + \theta_2 + \theta_3)$

In [19]:
```
zero_design = zero.subs(design)
zero_design = zero_design
zero_design
```

Out[19]: $\begin{bmatrix} 33.3\cos(\theta_1) + 54\cos(\theta_1 + \theta_2) + 13\cos(\theta_1 + \theta_2 + \theta_3) + 44 \\ 0 \end{bmatrix}$

In [20]:
```
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)
```

In [21]:
```
# THETA guesses
guess = numpy.array([-76,-126,-60,-81])*pi/180
guess
```

Out[21]:
```
array([-1.32645023, -2.19911486, -1.04719755, -1.41371669])
```

In [22]:
```
objective_function(guess)
```

Out[22]:
```
0.17882066438156152
```

In [23]:
```
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
```

In [24]:
```
points = sympy.Matrix([p3.T,p2.T,p1.T,p0.T,origin.T]).T
points
```

Out[24]: $\begin{bmatrix} l_0 + l_1\cos(\theta_1) - l_2\sin(\theta_1)\sin(\theta_2) + l_2\cos(\theta_1)\cos(\theta_2) + (-l_3\sin(\theta_2)\sin(\theta_3) + l_3\cos(\theta_2)\cos(\\ 0 \\ -l_1\sin(\theta_1) - l_2\sin(\theta_1)\cos(\theta_2) - l_2\sin(\theta_2)\cos(\theta_1) - (-l_3\sin(\theta_2)\sin(\theta_3) + l_3\cos(\theta_2)\cos(\theta \end{bmatrix}$

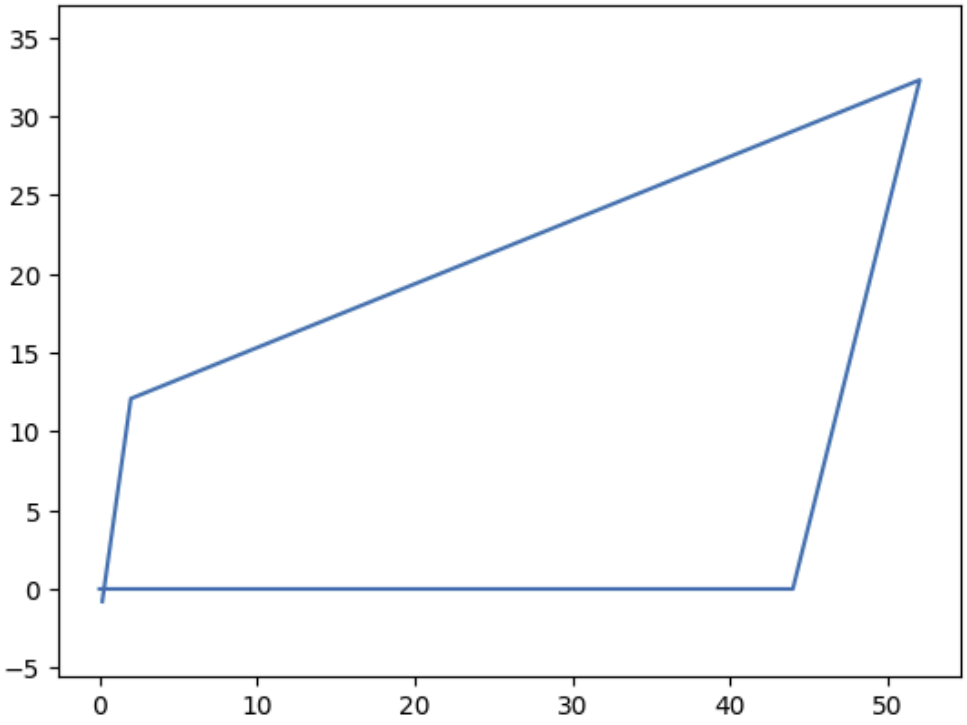
In [25]:
```
points_design = points.subs(design)
```

In [26]:
```
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')

plot_fourbar(guess) # FINAL MODEL
```

```
[[ 0.17882066  1.98807098 52.05599912 44.          0.        ]
 [ 0.          0.          0.          0.          0.        ]
 [-0.79139325 12.08209164 32.31084768  0.          0.        ]]
```



In [27]:
```
zero = sympy.Matrix(zero)
zero.simplify()
zero
```

Out[27]:
$$\begin{bmatrix} l_0 + l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2) + l_3 \cos(\theta_1 + \theta_2 + \theta_3) \\ 0 \end{bmatrix}$$

In [28]:
```
independent = sympy.Matrix([q1])
dependent = sympy.Matrix([q1,q2,q3,q4])
zero_design = zero.subs(design)
```

In [29]:
```
A = zero_design.jacobian(independent)
A
```

Out[29]:
$$\begin{bmatrix} -33.3 \sin(\theta_1) - 54 \sin(\theta_1 + \theta_2) - 13 \sin(\theta_1 + \theta_2 + \theta_3) \\ 0 \end{bmatrix}$$

In [30]:
```
B = zero_design.jacobian(dependent)
B
```

Out[30]:
$$\begin{bmatrix} -33.3\sin(\theta_1) - 54\sin(\theta_1+\theta_2) - 13\sin(\theta_1+\theta_2+\theta_3) & -54\sin(\theta_1+\theta_2) - 13\sin(\theta_1+\theta_2+\theta_3) \\ 0 & 0 \end{bmatrix}$$

In [31]:
```
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
```

Out[31]:
```
array([[-0.53848029],
       [ 0.        ]])
```

In [32]:
```
Bn = numpy.array(B.subs(state),dtype=float)
Bn
```

Out[32]:
```
array([[ -0.53848029, -32.84842135, -12.86117031,   0.        ],
       [  0.        ,   0.        ,   0.        ,   0.        ]])
```

In [33]:
```
print(Bn.shape)
```

```
(2, 4)
```

In [34]:
```
import numpy as np
Cn, *_ = np.linalg.lstsq(Bn, -An, rcond=None)
Cn
```

Out[34]:
```
array([[-0.00023295],
       [-0.01421063],
       [-0.0055639 ],
       [ 0.        ]])
```

In [35]:
```
p_out =v3_in_n
p_out
```

Out[35]:
$$\begin{bmatrix} (-l_3\sin(\theta_2)\sin(\theta_3) + l_3\cos(\theta_2)\cos(\theta_3))\cos(\theta_1) + (-l_3\sin(\theta_2)\cos(\theta_3) - l_3\sin(\theta_3)\cos(\theta_2)) \\ 0 \\ -(-l_3\sin(\theta_2)\sin(\theta_3) + l_3\cos(\theta_2)\cos(\theta_3))\sin(\theta_1) + (-l_3\sin(\theta_2)\cos(\theta_3) - l_3\sin(\theta_3)\cos(\theta_2)) \end{bmatrix}$$

In [36]:
```
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
```

Out[36]:
```
array([[-12.60385074],
       [  0.        ],
       [  1.85689791]])
```

In [37]:
```
y_dot = Jo@numpy.array([[1,]]).T
y_dot
```

```
Out[37]: array([[-12.60385074],
                [  0.        ],
                [  1.85689791]])

In [38]: p_out_n = numpy.array(p_out.subs(design).subs(state),dtype=float)
         p_out_n

Out[38]: array([[ -1.89480825],
                [  0.        ],
                [-12.86117031]])

In [ ]:
```

Another code for jacobian:

```python
import sympy as sp
from sympy import cos, sin, sqrt, atan2, simplify, diff

# Define symbols
theta1, theta2, theta3, theta4 = sp.symbols('theta1 theta2 theta3 theta4', real=True)
l1, l2, l3, l4, l_E = sp.symbols('l1 l2 l3 l4 l_E', positive=True)

print("="*70)
print("4-BAR LINKAGE - SYMBOLIC POSITION OF POINT E")
print("="*70)

# Closure equations (for reference)
print("\nClosure equations:")
eq1 = l2*cos(theta2) + l3*cos(theta3) - l1*cos(theta1) - l4*cos(theta4)
eq2 = l2*sin(theta2) + l3*sin(theta3) - l1*sin(theta1) - l4*sin(theta4)
print("Eq1:", eq1, "= 0")
print("Eq2:", eq2, "= 0")

# Position of point E
# Path: Origin -> Link 1 base -> Link 3 base -> Point E on Link 3
print("\n" + "="*70)
print("POSITION OF POINT E (SYMBOLIC)")
print("="*70)

print("\nPath to Point E:")
print("  Origin (0,0) -> Link 1 endpoint -> Along Link 3 by distance l_E")

# Link 1 endpoint (base of link 3)
x_link3_base = l1*cos(theta1)
y_link3_base = l1*sin(theta1)

print(f"\nLink 3 base position:")
print(f"  x_base = l1*cos(θ1)")
print(f"  y_base = l1*sin(θ1)")

# Point E is at distance l_E from link 3 base, along link 3 direction (θ3)
x_E = x_link3_base + l_E*cos(theta3)
y_E = y_link3_base + l_E*sin(theta3)
```

```
print(f"\nPoint E position:")
print(f"  x_E = l1*cos(θ1) + l_E*cos(θ3)")
print(f"  y_E = l1*sin(θ1) + l_E*sin(θ3)")

print("\nSimplified:")
print(f"  x_E = {x_E}")
print(f"  y_E = {y_E}")

# Since θ1 is fixed at 135°, we can express E purely in terms of θ3
print("\n" + "="*70)
print("WITH θ1 = 135° (FIXED)")
print("="*70)

theta1_val = sp.pi * 3 / 4  # 135° in radians

x_E_fixed = x_E.subs(theta1, theta1_val)
y_E_fixed = y_E.subs(theta1, theta1_val)

print(f"\nPosition with θ1 = 135°:")
print(f"  x_E = {x_E_fixed}")
print(f"  y_E = {y_E_fixed}")

# Now compute Jacobian: ∂E/∂θ2
# Since E depends on θ3, and θ3 depends on θ2, we use chain rule
print("\n" + "="*70)
print("JACOBIAN ∂E/∂θ2 (SYMBOLIC)")
print("="*70)

print("\nUsing chain rule:")
print("  ∂x_E/∂θ2 = ∂x_E/∂θ3 · ∂θ3/∂θ2")
print("  ∂y_E/∂θ2 = ∂y_E/∂θ3 · ∂θ3/∂θ2")

# Partial derivatives of E with respect to θ3
dx_E_dtheta3 = diff(x_E_fixed, theta3)
dy_E_dtheta3 = diff(y_E_fixed, theta3)

print(f"\n∂x_E/∂θ3 = {dx_E_dtheta3}")
print(f"∂y_E/∂θ3 = {dy_E_dtheta3}")
```

```
# For ∂θ3/∂θ2, we need to differentiate the constraint equations
print("\n" + "="*70)
print("FINDING ∂θ3/∂θ2 FROM CONSTRAINT EQUATIONS")
print("="*70)

print("\nDifferentiating closure equations with respect to θ2:")

# Differentiate eq1 and eq2 with respect to theta2
deq1_dtheta2 = diff(eq1, theta2) + diff(eq1, theta3)*diff(theta3, theta2) + diff(eq1, theta4)*diff(theta4, theta2)
deq2_dtheta2 = diff(eq2, theta2) + diff(eq2, theta3)*diff(theta3, theta2) + diff(eq2, theta4)*diff(theta4, theta2)

print("These give us a linear system in ∂θ3/∂θ2 and ∂θ4/∂θ2")

# Let's denote dtheta3_dtheta2 and dtheta4_dtheta2 as unknowns
dtheta3_dtheta2, dtheta4_dtheta2 = sp.symbols('dtheta3_dtheta2 dtheta4_dtheta2', real=True)

# Coefficients from differentiation
coeff_eq1_theta3 = -l3*sin(theta3)
coeff_eq1_theta4 = l4*sin(theta4)
coeff_eq1_theta2 = -l2*sin(theta2)

coeff_eq2_theta3 = l3*cos(theta3)
coeff_eq2_theta4 = -l4*cos(theta4)
coeff_eq2_theta2 = l2*cos(theta2)

print(f"\nFrom Eq1: {coeff_eq1_theta3}·∂θ3/∂θ2 + {coeff_eq1_theta4}·∂θ4/∂θ2 = {-coeff_eq1_theta2}")
print(f"From Eq2: {coeff_eq2_theta3}·∂θ3/∂θ2 + {coeff_eq2_theta4}·∂θ4/∂θ2 = {-coeff_eq2_theta2}")

# Solve the linear system
sol = sp.solve([
    coeff_eq1_theta3*dtheta3_dtheta2 + coeff_eq1_theta4*dtheta4_dtheta2 + coeff_eq1_theta2,
    coeff_eq2_theta3*dtheta3_dtheta2 + coeff_eq2_theta4*dtheta4_dtheta2 + coeff_eq2_theta2
], [dtheta3_dtheta2, dtheta4_dtheta2])

print("\n" + "="*70)
print("SOLUTION FOR ∂θ3/∂θ2")
print("="*70)
print(f"\n∂θ3/∂θ2 = {simplify(sol[dtheta3_dtheta2])}")
```

```
# Final Jacobian
dx_E_dtheta2 = simplify(dx_E_dtheta3 * sol[dtheta3_dtheta2])
dy_E_dtheta2 = simplify(dy_E_dtheta3 * sol[dtheta3_dtheta2])

print("\n" + "="*70)
print("FINAL JACOBIAN EQUATIONS")
print("="*70)
print(f"\n∂x_E/∂θ2 = {dx_E_dtheta2}")
print(f"\n∂y_E/∂θ2 = {dy_E_dtheta2}")

print("\n" + "="*70)
print("SUMMARY")
print("="*70)
print("\nPosition of Point E:")
print(f"  x_E = {x_E_fixed}")
print(f"  y_E = {y_E_fixed}")
print("\nJacobian components:")
print(f"  ∂x_E/∂θ2 = {dx_E_dtheta2}")
print(f"  ∂y_E/∂θ2 = {dy_E_dtheta2}")
print("="*70)
```

✓ 1.0s

```
======================================================================
4-BAR LINKAGE - SYMBOLIC POSITION OF POINT E
======================================================================

Closure equations:
Eq1: -l1*cos(theta1) + l2*cos(theta2) + l3*cos(theta3) - l4*cos(theta4) = 0
Eq2: -l1*sin(theta1) + l2*sin(theta2) + l3*sin(theta3) - l4*sin(theta4) = 0

======================================================================
POSITION OF POINT E (SYMBOLIC)
======================================================================

Path to Point E:
  Origin (0,0) -> Link 1 endpoint -> Along Link 3 by distance l_E

Link 3 base position:
  x_base = l1*cos(θ1)
  y_base = l1*sin(θ1)

Point E position:
  x_E = l1*cos(θ1) + l_E*cos(θ3)
  y_E = l1*sin(θ1) + l_E*sin(θ3)

Simplified:
  x_E = l1*cos(theta1) + l_E*cos(theta3)
  y_E = l1*sin(theta1) + l_E*sin(theta3)

======================================================================
WITH θ1 = 135° (FIXED)
======================================================================

Position with θ1 = 135°:
  x_E = -sqrt(2)*l1/2 + l_E*cos(theta3)
  y_E = sqrt(2)*l1/2 + l_E*sin(theta3)

======================================================================
JACOBIAN ∂E/∂θ2 (SYMBOLIC)
======================================================================

Using chain rule:
  ∂x_E/∂θ2 = ∂x_E/∂θ3 · ∂θ3/∂θ2
  ∂y_E/∂θ2 = ∂y_E/∂θ3 · ∂θ3/∂θ2
```

```
∂x_E/∂θ3 = -l_E*sin(theta3)
∂y_E/∂θ3 = l_E*cos(theta3)

======================================================================
FINDING ∂θ3/∂θ2 FROM CONSTRAINT EQUATIONS
======================================================================

Differentiating closure equations with respect to θ2:
These give us a linear system in ∂θ3/∂θ2 and ∂θ4/∂θ2

From Eq1: -l3*sin(theta3)·∂θ3/∂θ2 + l4*sin(theta4)·∂θ4/∂θ2 = l2*sin(theta2)
From Eq2: l3*cos(theta3)·∂θ3/∂θ2 + -l4*cos(theta4)·∂θ4/∂θ2 = -l2*cos(theta2)

======================================================================
SOLUTION FOR ∂θ3/∂θ2
======================================================================

∂θ3/∂θ2 = -l2*sin(theta2 - theta4)/(l3*sin(theta3 - theta4))

======================================================================
FINAL JACOBIAN EQUATIONS
======================================================================

∂x_E/∂θ2 = l2*l_E*sin(theta3)*sin(theta2 - theta4)/(l3*sin(theta3 - theta4))

∂y_E/∂θ2 = -l2*l_E*sin(theta2 - theta4)*cos(theta3)/(l3*sin(theta3 - theta4))
```

```
======================================================================
SUMMARY
======================================================================

Position of Point E:
  x_E = -sqrt(2)*l1/2 + l_E*cos(theta3)
  y_E = sqrt(2)*l1/2 + l_E*sin(theta3)

Jacobian components:
  ∂x_E/∂θ2 = l2*l_E*sin(theta3)*sin(theta2 - theta4)/(l3*sin(theta3 - theta4))
  ∂y_E/∂θ2 = -l2*l_E*sin(theta2 - theta4)*cos(theta3)/(l3*sin(theta3 - theta4))
======================================================================
```

7. Discussion

```
======================================================================
GRASSHOPPER ROBOT LEG - JACOBIAN FORCE & VELOCITY ANALYSIS
======================================================================

Leg Geometry:
  l₁ = 13 mm (ground link, fixed at 135°)
  l₂ = 54 mm (input/actuated link)
  l₃ = 33 mm (coupler)
  l₄ = 44 mm (follower)
  l₅ = 36 mm (foot extension - one piece with l₃)
  Total reach: l₃ + l₅ = 69 mm

======================================================================
1. BIOMECHANICS - GRASSHOPPER JUMPING FORCES
======================================================================

Real Grasshopper Data:
  - Body mass: 2-3 grams
  - Jump height: 0.5-1.0 meters
  - Takeoff velocity: 3-4 m/s
  - Peak GRF: 5-15× body weight
  - Leg force duration: 20-30 ms

Robot Specifications:
  Total mass: 50 grams
  Body weight: 0.491 N
```

```
Jump Performance Targets:
  Target jump height: 30 cm
  Required takeoff velocity: 2.43 m/s
  Jump duration: 25 ms
  Average acceleration: 97.0 m/s² (9.9g)

======================================================================
GROUND REACTION FORCE (GRF) ESTIMATES
======================================================================

1. Static Standing (per leg, 6 legs):
   F_static = 0.082 N

2. Walking (per leg):
   F_walk = 0.123 N

3. Jump Preparation (per hind leg, 2 legs):
   F_prep = 0.245 N

4. Peak Jumping Force (per hind leg):
   F_jump = 2.671 N
   = 10.9× body weight per leg

5. Maximum Design Force (safety factor 1.5×):
   F_max = 4.007 N
```


```
======================================================================
GROUND REACTION FORCE (GRF) ESTIMATES
======================================================================

1. Static Standing (per leg, 6 legs):
   F_static = 0.082 N

2. Walking (per leg):
   F_walk = 0.123 N

3. Jump Preparation (per hind leg, 2 legs):
   F_prep = 0.245 N

4. Peak Jumping Force (per hind leg):
   F_jump = 2.671 N
   = 10.9× body weight per leg

5. Maximum Design Force (safety factor 1.5×):
   F_max = 4.007 N
```

```
======================================================================
2. FORCE TRANSMISSION ANALYSIS - JUMP CYCLE
======================================================================

======================================================================
Configuration: Crouch (stance) (θ₂ = -90°)
======================================================================

Kinematics:
  θ₂ = -90.0°  (input angle)
  θ₃ =  138.23°  (coupler angle)
  θ₄ =  249.49°  (follower angle)
  ∂θ₃/∂θ₂ = 0.6153
  Foot position: (-60.65, 1.16) mm

Jacobian J:
  [∂x/∂θ₂]   [   25.7174] mm/rad
  [∂y/∂θ₂] = [  -31.6654] mm/rad
  ||J|| = 40.7932 mm/rad

----------------------------------------------------------------------
TORQUE REQUIREMENTS (τ = J^T × F):
----------------------------------------------------------------------
  Static/leg  : F =  0.082 N  →  τ = -0.00259 N·m =   -2.59 mN·m
  Walking     : F =  0.123 N  →  τ = -0.00388 N·m =   -3.88 mN·m
  Jump prep   : F =  0.245 N  →  τ = -0.00777 N·m =   -7.77 mN·m
  Peak jump   : F =  2.671 N  →  τ = -0.08459 N·m =  -84.59 mN·m
  Maximum     : F =  4.007 N  →  τ = -0.12688 N·m = -126.88 mN·m
```

```
======================================================================
Configuration: Mid-extension (θ₂ = -70°)
======================================================================

Kinematics:
  θ₂ = -70.0°  (input angle)
  θ₃ =  151.09°  (coupler angle)
  θ₄ =  268.40°  (follower angle)
  ∂θ₃/∂θ₂ = 0.6779
  Foot position: (-51.13, -8.19) mm

Jacobian J:
  [∂x/∂θ₂]   [   28.1321] mm/rad
  [∂y/∂θ₂] = [  -22.4759] mm/rad
  ||J|| = 36.0081 mm/rad

----------------------------------------------------------------------
TORQUE REQUIREMENTS (τ = J^T × F):
----------------------------------------------------------------------
  Static/leg  : F =  0.082 N  →  τ = -0.00184 N·m =   -1.84 mN·m
  Walking     : F =  0.123 N  →  τ = -0.00276 N·m =   -2.76 mN·m
  Jump prep   : F =  0.245 N  →  τ = -0.00551 N·m =   -5.51 mN·m
  Peak jump   : F =  2.671 N  →  τ = -0.06004 N·m =  -60.04 mN·m
  Maximum     : F =  4.007 N  →  τ = -0.09006 N·m =  -90.06 mN·m
```

```
======================================================================
Configuration: Full extension (θ₂ = -50°)
======================================================================

Kinematics:
  θ₂ = -50.0°  (input angle)
  θ₃ =  165.59°  (coupler angle)
  θ₄ =  285.75°  (follower angle)
  ∂θ₃/∂θ₂ = 0.7774
  Foot position: (-41.31, -15.01) mm

Jacobian J:
  [∂x/∂θ₂]   [   28.0219] mm/rad
  [∂y/∂θ₂] = [  -17.2413] mm/rad
  ||J|| = 32.9012 mm/rad

----------------------------------------------------------------------
TORQUE REQUIREMENTS (τ = J^T × F):
----------------------------------------------------------------------
  Static/leg  : F =  0.082 N  →  τ = -0.00141 N·m =   -1.41 mN·m
  Walking     : F =  0.123 N  →  τ = -0.00211 N·m =   -2.11 mN·m
  Jump prep   : F =  0.245 N  →  τ = -0.00423 N·m =   -4.23 mN·m
  Peak jump   : F =  2.671 N  →  τ = -0.04606 N·m =  -46.06 mN·m
  Maximum     : F =  4.007 N  →  τ = -0.06909 N·m =  -69.09 mN·m
```

```
======================================================================
3. VELOCITY ANALYSIS - JUMPING KINEMATICS
======================================================================

Biomechanics velocity data:
  - Grasshopper leg extension velocity: 1-3 m/s during jump
  - Foot velocity at takeoff: 3-4 m/s
  - Extension time: 20-30 ms

Target Foot Velocities:
  Slow (walking):   500 mm/s = 0.5 m/s
  Medium (fast):   1500 mm/s = 1.5 m/s
  Fast (jumping):  3000 mm/s = 3.0 m/s

======================================================================
Crouch (stance) (θ₂ = -90°)
======================================================================

For VERTICAL foot motion (v_y):
Speed           v_y (mm/s)   ω₂ (rad/s)   ω₂ (deg/s)   ω₂ (RPM)
----------------------------------------------------------------------
Slow            500          -15.790      -904.7       -150.8
Medium          1500         -47.370      -2714.1      -452.4
Fast            3000         -94.741      -5428.2      -904.7
```

```
======================================================================
4. WORKSPACE SWEEP - JACOBIAN ACROSS ENTIRE RANGE
======================================================================

    θ₂       θ₃     ||J||      τ@jump      ω@3m/s    Foot X    Foot Y
 (deg)    (deg) (mm/rad)     (mN·m)       (RPM)     (mm)      (mm)
----------------------------------------------------------------------
-120.0   120.19    49.15    -128.14      -597.2   -70.89     22.07
-114.3   123.62    47.46    -120.43      -635.5   -69.61     17.43
-108.6   127.04    45.82    -112.18      -682.2   -67.95     13.08
-102.9   130.45    44.23    -103.64      -738.4   -65.97      9.05
 -97.1   133.88    42.68     -95.05      -805.1   -63.73      5.35
 -91.4   137.35    41.16     -86.64      -883.3   -61.29      1.96
 -85.7   140.89    39.70     -78.62      -973.4   -58.69     -1.13
 -80.0   144.50    38.28     -71.19     -1075.0   -55.99     -3.92
 -74.3   148.22    36.95     -64.50     -1186.6   -53.22     -6.45
 -68.6   152.06    35.71     -58.67     -1304.4   -50.42     -8.75
 -62.9   156.04    34.62     -53.77     -1423.2   -47.61    -10.84
 -57.1   160.18    33.71     -49.81     -1536.5   -44.81    -12.78
 -51.4   164.49    33.03     -46.71     -1638.4   -42.01    -14.57
 -45.7   168.98    32.62     -44.35     -1725.7   -39.21    -16.27
 -40.0   173.66    32.50     -42.52     -1799.9   -36.40    -17.89
```

```
======================================================================
5. MOTOR/ACTUATOR SPECIFICATIONS FOR GRASSHOPPER ROBOT
======================================================================

✓ TORQUE REQUIREMENTS:
  Minimum (across workspace):  -128.14 mN·m = -0.12814 N·m
  Maximum (across workspace):  -42.52 mN·m = -0.04252 N·m
  Recommended (with 20% margin): -51.02 mN·m = -0.05102 N·m

✓ SPEED REQUIREMENTS (for 3 m/s jumping):
  Minimum: -1799.9 RPM = -10799.3 deg/s
  Maximum: -597.2 RPM = -3583.5 deg/s
  Recommended (with 20% margin): -716.7 RPM

✓ POWER REQUIREMENTS:
  Peak power: 2.659 W = 2659.3 mW
  Recommended: 3.989 W (with 50% margin)

✓ SUGGESTED MOTOR TYPES:
  - Micro servo (e.g., TowerPro SG90): 1.8 kg·cm = 0.177 N·m ✓
  - Mini servo (e.g., MG90S): 2.2 kg·cm = 0.216 N·m ✓
  - Brushless micro motor with gearbox (higher efficiency)

✓ JACOBIAN CONDITIONING:
  Range: 32.50 - 49.15 mm/rad
  Best configuration: θ₂ = -120.0° (||J|| = 49.15)
  Worst configuration: θ₂ = -40.0° (||J|| = 32.50)
  Condition variation: 1.51× (good if < 3×)
```

8.1 Degrees of freedom and motors

- **Per leg linkage:** planar 4-bar (or equivalent) with **2 DOF**. (2 motors)
- **Two legs coupled** on a common shaft → **2 actuated DOF total** for cocking.
- **Latch** adds a binary state (locked/unlocked). Release can be via the same servo (cam/tab) or a tiny solenoid/micro-servo (not counted as a continuous DOF).
- **Remaining states** (tibia extension during jump, body pitch) are **determined passively** by geometry, stored energy, and ground contact—no continuous actuation needed.
- **Motors:** 1× SG90 to preload **both legs** (or 2× SG90, one per leg, for margin). We arrived at "1 DOF actuation" by constraining both legs to the same preload shaft (mechanical coupling) and using a latch to time the release.

8.2 How we estimated end-effector forces

- From literature, peak vertical GRF is **~6–10× body weight**.
- Using our mass m=0.0025 kgm=0.0025 kg, take-off time t≈0.05 st≈0.05 s, and vertical velocity component vyvy:
  ay=vy/tay=vy/t and F=m(ay+g)F=m(ay+g) → **0.14–0.18 N** (angle-dependent: ~0.14 N at 45°, ~0.18 N at 72.6°).
- This force times the joint moment arm gave **joint torque** used for sizing the spring and the cocking servo.

8.3 How we estimated end-effector speeds

- We decomposed measured take-off speed v=3.24 m/sv=3.24 m/s into components: vy=vsinθ, vx=vcosθvy=vsinθ, vx=vcosθ.
    - At **72.6°**: vy≈3.09 m/svy≈3.09 m/s, vx≈0.95 m/svx≈0.95 m/s.
    - At **45°**: vy=vx≈2.29 m/svy=vx≈2.29 m/s.
- For leg tip speed we used **Chen et al.** timing: rapid femur–tibia extension of ~100◦100◦ (≈1.75≈1.75 rad) over ~15 ms → ω≈116 rad/sω≈116 rad/s. With tibia length L≈22 mmL≈22 mm, tip speed vtip≈ωL≈2.5 m/svtip≈ωL≈2.5 m/s, consistent with COM speed.

References

Chen, DS., Yin, JM., Chen, KW. *et al.* Biomechanical and dynamic mechanism of locust take-off. *Acta Mech Sin* 30, 762–774 (2014). https://doi.org/10.1007/s10409-014-0065-2

Hawlena, D., Kress, H., Dufresne, E.R. and Schmitz, O.J. (2011), Grasshoppers alter jumping biomechanics to enhance escape performance under chronic risk of spider predation. Functional Ecology, 25: 279-288. https://doi.org/10.1111/j.1365-2435.2010.01767.x https://besjournals.onlinelibrary.wiley.com/action/showCitFormats?doi=10.1111%2Fj.1365-2435.2010.01767.x

Konez Eroğlu, A. (2007). *Development and Analysis of Grasshopper-Like Jumping Mechanism in Biomimetic Approach* (Order No. 31669257). Available from ProQuest Dissertations & Theses Global. (3122726176). https://login.ezproxy1.lib.asu.edu/login?url=https://www.proquest.com/dissertations-theses/development-analysis-grasshopper-like-jumping/docview/3122726176/se-2

Malcolm Burrows, Gregory P. Sutton; Locusts use a composite of resilin and hard cuticle as an energy store for jumping and kicking. *J Exp Biol* 1 October 2012; 215 (19): 3501–3512. doi: https://doi.org/10.1242/jeb.071993

Zhang, ZQ., Yang, Q., Zhao, J. *et al.* Dynamic model and performance analysis of rigid-flexible coupling four-bar leg mechanism for small scale bio-inspired jumping robot. *Microsyst Technol* 25, 3269–3285 (2019). https://doi.org/10.1007/s00542-019-04546-5