1.5em 0pt

Mini Project Report
ME763

Design of the compliant gripper and pipe flow control

Course Instructor :
Prof. Bishakh Bhattacharya

Project Mentor :
Dr. Suraj Kumar Mishra

Submitted by :

Abhishek Kumar(220043)
Avinash Bhashkar(241050632)
Raj Kumar(241050620)

Mechanical Department :
Indian Institute of Technology Kanpur

Contents

List of Figures

1. Introduction

Robotic grippers are crucial for automating tasks across diverse fields, from intricate manufacturing processes to delicate surgical procedures, and even in the exploration of extreme environments. These manipulators are designed to interact with their surroundings and manipulate objects with increasing levels of precision, adaptability, and robustness. The development of robotic grippers is driven by the need for effective handling in situations where human intervention is either impractical or impossible, and is characterized by the integration of advanced materials, sophisticated actuation methods, and intelligent control systems. The evolution of gripper technology has been marked by a shift from traditional rigid designs to more versatile soft grippers that mimic the dexterity of biological systems, particularly the human hand. This shift is not merely about replicating biological form, but about achieving functional advantages that can enhance performance in a variety of applications.

1.A. Motivation

The motivation for this project arises from the challenges faced in current intravenous (IV) drug infusion systems, which rely heavily on manual intervention. These challenges are summarized below:

Need for Frequent Monitoring Regular, direct observation is required to check medicine levels and ensure proper flow regulation. This increases the burden on healthcare staff, where a nurse is manually monitoring an IV system.

Manual Flow Adjustment Existing systems rely on manual intervention to adjust roller clamps for controlling the flow of medicine. This approach leads to potential inconsistencies and inefficiencies which shows a typical roller clamp mechanism. The limitations of current IV flow regulation systems highlight the need for an automated mechanism that reduces manual intervention, ensures consistent medicine delivery, and alleviates the workload on healthcare professionals. The proposed solution aims to address these challenges by incorporating precise flow control, compact design, adjustable gripping force, and long-term durability.

Flow Control Requirements To address these challenges, the proposed solution must meet specific requirements: Flow Control Range: The mechanism should regulate flow rates between 10 mL/hour to 500 mL/hour, which is typical for IV drug infusion. Gripping Force: The mechanism should provide an adjustable force to pinch tubes with inner diameters of 2–4 mm without causing permanent deformation. Gripper Assembly Size: A compact design with dimensions less than 50 mm $\times$ 50 mm $\times$ 50 mm is necessary for compatibility with standard IV setups. Durability: The mechanism must withstand at least 100,000 cycles of operation without mechanical failure.

1.B. Literature review

The field of robotic grippers is a convergence of multiple disciplines, including mechanical engineering, materials science, and control systems. Traditional grippers often rely on rigid links and joints, which can be effective for many tasks but may lack the adaptability needed for handling irregularly shaped or fragile objects. To overcome these limitations, compliant mechanisms and soft robotics have emerged as viable alternatives. Compliant mechanisms use flexible elements rather than rigid joints, offering advantages in terms of accuracy and eliminating the errors from the assembly of many links and joints. Soft grippers, on the other hand, use materials and designs that allow them to deform and conform to the shape of the object being grasped, providing a more secure and gentle grip. The design process for these grippers involves careful consideration of multiple factors, including material properties, actuation principles, optimization of the design, and the specific requirements of the intended application.

The design, actuation, materials, fabrication, and control of soft robotic grippers have been extensively studied in recent years. Inspired by nature, many gripper designs replicate the human hand, twining action of climbing plants, or Fin Ray structures to achieve adaptive grasping [4,

8, 6]. Compliant mechanisms are commonly used in gripper designs, offering stiffness for holding objects while maintaining flexibility for motion generation. These mechanisms can be monolithic structures created at various length scales without requiring assembly [3, 2]. Underactuated mechanisms simplify control and allow adaptation to objects of different shapes [7]. Tunable stiffness methods enable grippers to switch between soft and hard configurations, facilitating handling of irregular geometries while achieving high holding forces [8, 1].

Actuation techniques include tendon-driven systems for intricate finger movements [13], fluidic elastomer actuators (FEAs) that deform through inflation [12], and pneumatic systems combined with granular jamming for enhanced stiffness control [11]. Shape memory alloy (SMA) wires are widely used for compact designs with displacement amplification capabilities [2]. Piezoelectric actuators are employed in high-precision microgrippers [10], while dielectric elastomer actuators (DEAs) and ionic polymer-metal composites (IPMCs) provide alternative actuation methods with varying trade-offs in stress generation and response speed [9].

Sensing and control play a crucial role in enhancing the functionality of soft robotic grippers. Open-loop and closed-loop control schemes are used depending on application requirements. Sliding mode control is effective for precise closed-loop control [2]. Self-sensing techniques measure resistance changes in SMA wires or strain-sensitive conductive filaments with carbon nanotubes to detect force direction and magnitude [14, 2]. Optical methods like lossy waveguides and tactile sensors fabricated using conductive filaments further enhance sensing capabilities [14]. Performance metrics for soft robotic grippers include holding force, displacement amplification ratio, natural frequency, safety factor, and motion resolution. Grippers must adapt to objects of varying shapes and sizes while maintaining durability under repeated use. For example, Mousavi et al. (2022) achieved a gauge factor of 1342 using FFF with conductive filaments for multidirectional tactile sensing [14], while Then Mozhi et al. (2023) designed a monolithic compliant gripper capable of handling objects up to 35 g with a displacement amplification ratio of 3.7 [2].

2. Proposed Design and mathematical modeling

2.A. Cad model and geometry

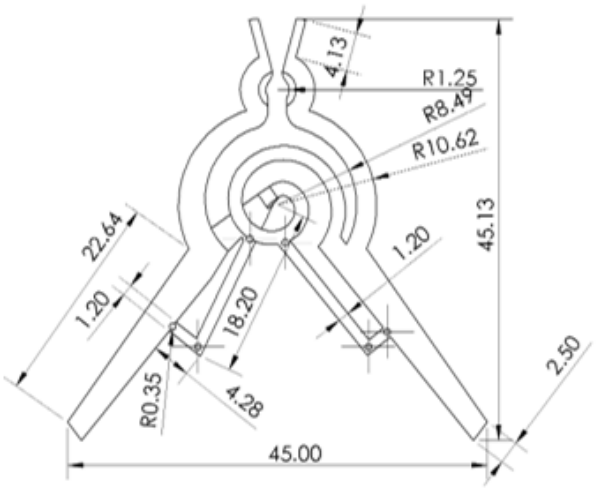


Figure 1: Geometric drawing of gripper

The CAD model and geometry of the proposed mechanism are presented to highlight its design, material specifications, and innovative features. In Figure 2, the rigid body CAD model is depicted with precise dimensions, showcasing the compact structure of the mechanism with overall dimensions of $46 \times 46 \times 11$ mm. The input force and output force are applied horizontally, as illustrated. The revolute joints (K1 to K6) allow rotational motion, while the central revolute

joint coordinates the movement of all connected links. The lightweight design is achieved using PLA+ material, with a total weight of just 3 grams.

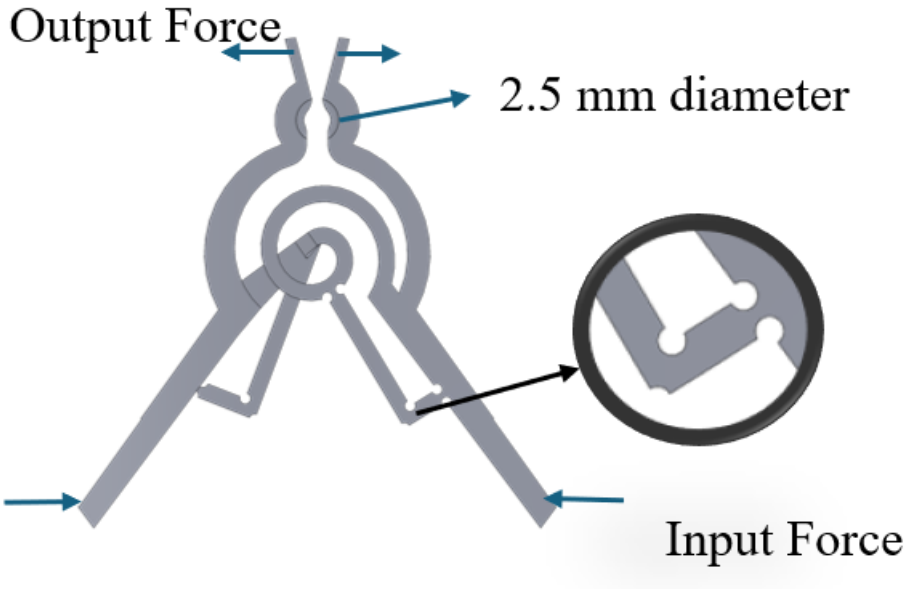


Figure 2: Cad model of rigid and complaint mechanism

Figure 3 shows the CAD model model of rgid and complaint mechanism, which simplifies the visualization of force transmission through the mechanism. This representation highlights how input forces are transmitted to generate output forces via the compliant elements and rigid links. Additionally, Figure 2 provides isometric views of the CAD model, offering a three-dimensional perspective that emphasizes the spatial arrangement of components.

The mechanism incorporates a spiral spring as a compliant element that also serves as an energy-storing component. This design enhances durability and ensures a higher life cycle compared to conventional mechanisms. The simplicity and precision of the mechanism make it particularly suitable for applications such as intravenous drug infusion systems. Furthermore, as per our understanding, no prior studies have documented such a compliant mechanism in existing literature, emphasizing its novelty. Finally, Figure 3 demonstrates an visual representation of the CAD model's functionality. This visual representation explains how input forces are transmitted through the links and springs to generate output forces, providing insight into the dynamic behavior of the mechanism under applied loads.

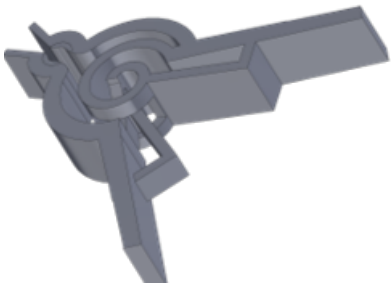


Figure 3: Isometric view of the CAD model

Stiffness calculation of Flexure Revolute Joint

The stiffness of the flexure revolute joint is derived based on its geometry and material properties. The key stiffness components include torsional stiffness (k_t), bending stiffness (k_b), and axial stiffness (k_a).

On-axis stiffness $\mathrm{N}\cdot\mathrm{mm/rad}\ \times 10^{-3}$			Off-axis STIFFNESS $\mathrm{N/mm}\ \times 10^{-3}$		
$K_{\theta z}$	$K_{\theta y}$	$K_{\theta x}$	K_z	K_y	K_x
6.481	648.15	246.1	24	0.24	77.78
0.112	—	—	—	—	—
23.342	1267.6	485.33	55.05	1.153	152.11
20.43	—	—	—	15.53	265.04
7.103	—	—	—	77.4	164.64
23.342	1267.6	—	—	57.186	152.11
36.12	1832	—	80.66	1.758	109.84
34.142	1585.1	—	68.15	1.642	94.99

Table 1: Comparison of on-axis and off-axis stiffness values for different joints [5]

Compliant revolute joint	b (mm)	L, l_f (mm)	t, r (mm)	C, R, g, h_f (mm)
Figs. **??**(a)–(d)	10	9	1	4.5

Table 2: Geometric parameters values for compliant revolute joints [5]

Torsional Stiffness (k_t)

Objective

The aim is to calculate the spring constant K (also called torsional stiffness) for a rectangular spring element using its material and geometric properties.

Theory and Formula

The spring constant K for a torsional spring is given by:

$$K = \frac{G \cdot J}{L}$$

where:

- G is the shear modulus of the material (a measure of rigidity),
- J is the torsional moment of inertia (depends on cross-section shape),
- L is the length of the spring.

The torsional moment of inertia J for a rectangular cross-section is:

$$J = \frac{b \cdot h^3}{12}$$

where:

- b is the width of the cross-section,
- h is the height (or thickness) of the cross-section.

This formula arises from mechanics of materials and considers how the shape resists twisting.

Given Values

- Width, $b = 1.5\,\text{mm} = 1.5 \times 10^{-3}\,\text{m}$
- Height, $h = 8\,\text{mm} = 8 \times 10^{-3}\,\text{m}$
- Length, $L = 31\,\text{mm} = 31 \times 10^{-3}\,\text{m}$
- Young's Modulus, $E = 3.5 \times 10^{9}\,\text{Pa}$
- Poisson's Ratio, $\nu = 0.3$

Step 1: Moment of Inertia Calculation

We substitute the values into the formula for J:

$$J = \frac{1.5 \times 10^{-3} \cdot (8 \times 10^{-3})^3}{12} = 6.4 \times 10^{-12}\,\text{m}^4$$

This value tells us how resistant the rectangular cross-section is to twisting.

Step 2: Shear Modulus Calculation

The shear modulus G is calculated using the relation:

$$G = \frac{E}{2(1+\nu)} = \frac{3.5 \times 10^{9}}{2(1+0.3)} = 1.35 \times 10^{9}\,\text{Pa}$$

This relation comes from basic elasticity theory, where E and ν define the material's response to stress.

Step 3: Spring Constant Calculation

Now, we plug in all values into the original formula for K:

$$K = \frac{G \cdot J}{L} = \frac{1.35 \times 10^{9} \cdot 6.4 \times 10^{-12}}{31 \times 10^{-3}} = 2.78\,\text{N/m}$$

So, the spring constant is approximately:

$$\boxed{K \approx 2.78\,\text{N/m}}$$

This value represents how much force per unit displacement the spring resists when twisted.

References

- Timoshenko, S., & Goodier, J. N. (1970). *Theory of Elasticity* (3rd ed.). McGraw-Hill Education.
- Khaururujal, K. (2011). The dependence of the spring constant in the linear range on spring parameters. *Physics Education, 46*(5), 540–544.

2.A. Kinematics and Force-Deflection Analysis

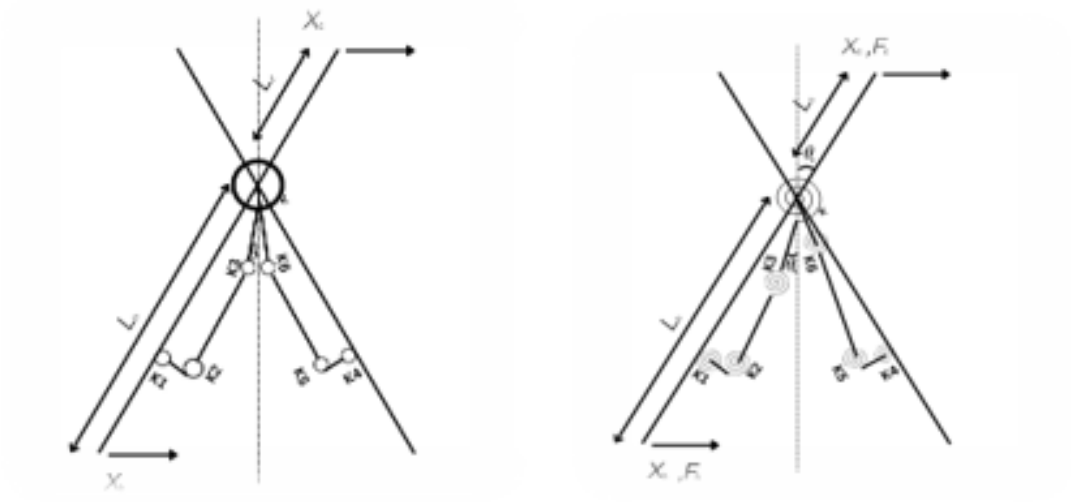


Figure 4: Kinematic representation of the gripper mechanism(left) and Pseudo-rigid body model(right)

The kinematics and force-deflection analysis of the mechanism are described using the rigid body CAD model and its pseudo-rigid body representation, as shown in Figure 4. The equations below detail the relationships between input forces, output forces, displacements, and deflections.

Forward Kinematics (FK): The forward kinematics equation relates the horizontal displacement x_1 to the angular displacement θ of a link with length L. For small angular displacements:

$$x_1 = L_1 \sin(\theta) \approx L_1\theta \tag{1}$$

Inverse Kinematics (IK): The inverse kinematics equation provides the angular displacement required to achieve a given horizontal displacement:

$$\theta = \frac{x_1}{L_1} \tag{2}$$

Force-Deflection Relationship Here, Let the spring constant of the torsional spring be K_0, So

$$F_1 L_1 = K_0\theta$$

$$\theta = \frac{F_1 L_1}{K_0}$$

Now let the other end displaced by x_2, So,

$$x_2 = L_2 \times \theta = \frac{F_1 L_1 L_2}{K_0}$$

$$x_2 = \frac{F_1 L_1 L_2}{K_0}$$

So, Required force to deflect the other end by x,

$$F_1 = F_{req} = \frac{K_0 x_2}{L_1 L_2}$$

The relationship between input displacement (X_1) and output displacement (X_2) is expressed

as:

$$X_2 = X_1 \cdot \frac{L_2}{L_1} \tag{3}$$

Here, $L_1, andL_2$ represent the lengths of the input and output links and their respective angular displacements.

Similarly, the input force (F_1) and output force (F_2) are related by:

$$F_2 = F_1 \cdot \frac{L_1}{L_2} \tag{4}$$

Equations (1) to 5 collectively describe how input forces, displacements, and deflections are transformed into corresponding outputs through the mechanism's geometry, material properties, and compliant elements. These relationships provide a foundation for analyzing performance under various loading conditions.

Where:

- F_1, F_2 : Input and output forces.
- X_2, X_2 : Input and output displacements.
- L_1, L_2 : Lengths of the input and output links.
- k_0, EI : Spring constant and flexural rigidity.
- θ_h, θ_c : Angular displacements of input and output links.
- x_1 : Deflection caused by applied force.
- F_{in} : Required input force.
- L_1, L_2 : Lengths of a link.
- $\sin(\theta)$: Sine function for angular displacement.

2.B. Kineto-static modelling

Moment Balance at Pivot: The moment balance equation at the pivot is given by:

$$M = \sum \tau = \tau_s + (F_{\text{in}} L_1) = 0 \tag{5}$$

Substituting the spring torques and forces:
-k$\theta_h - k_1\theta_1 - k_2\theta_2 - k_3\theta_3 + F_{\text{in}}L_1 = 0$
Solving for F_{in}:

$\text{F}_{\text{in}} = \frac{k\theta_h + k_1\theta_1 + k_2\theta_2 + k_3\theta_3}{L_1}$ (6)

Here, $k_\theta, \theta_h, \theta_1, \theta_2, \theta_3$ are variables representing spring constants and angular displacements.

2.C. Finite element analysis

Applied Load and Boundary Conditions A static force of **1.4142 N** was applied at point A, while point B was fixed to simulate real-world constraints. These boundary conditions were consistent across both configurations to ensure a fair comparison.

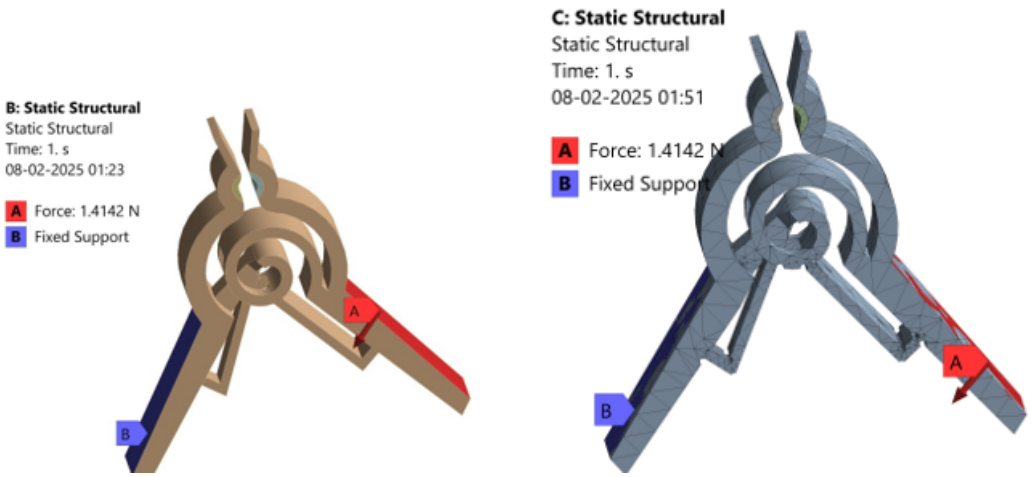


Figure 5: Boundary conditions

The Finite Element Analysis (FEA) was conducted to evaluate the structural performance of the mechanism under static loading conditions. The analysis includes simulations for total deformation, equivalent stress (von-Mises stress), and safety factor for two configurations: B: Static Structural and C: Static Structural. The results are summarized below.

Total Deformation The total deformation of the mechanism was analyzed to understand the displacement of components under the applied load:

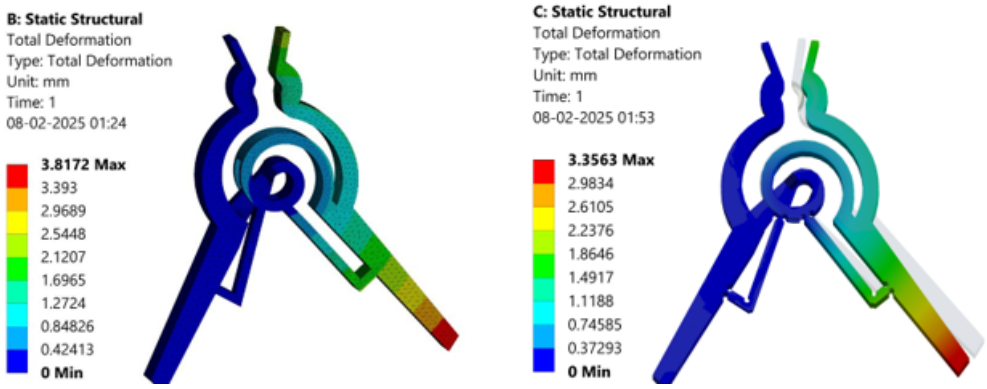


Figure 6: Total deformation

- For B: Static Structural, the maximum deformation is **3.8172 mm**, as shown in the second image of the top row. The deformation is concentrated near the loaded end of the mechanism.
- For C: Static Structural, the maximum deformation is **3.3563 mm**, as shown in the second image of the bottom row. This indicates slightly higher flexibility in configuration C.

Equivalent Stress (von-Mises Stress) The equivalent stress distribution highlights areas experiencing high stress:

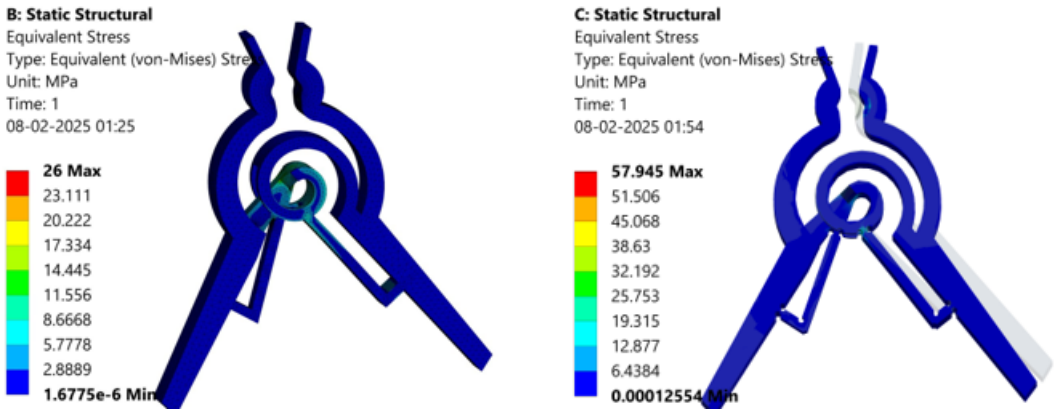


Figure 7: Equivalent (von-Mises) stress

- For B: Static Structural, the maximum von-Mises stress is **26 MPa**, as shown in the third image of the top row. The stress is concentrated around the pivot and near the applied load.
- For C: Static Structural, the maximum von-Mises stress increases to **57.945 MPa**, as shown in the third image of the bottom row, indicating higher stress under similar loading conditions.

Safety Factor The safety factor analysis provides an indication of how much stronger the design is compared to the applied load:

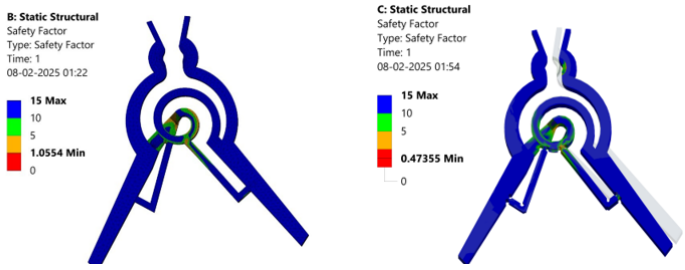


Figure 8: SOF

- For B: Static Structural, the minimum safety factor is **1.0554**, as shown in the fourth image of the top row, indicating that this configuration is just above failure threshold at critical points.
- For C: Static Structural, the minimum safety factor drops to **0.47355**, as shown in the fourth image of the bottom row, indicating potential failure at critical points under similar loading conditions.

3. Fatigue analysis

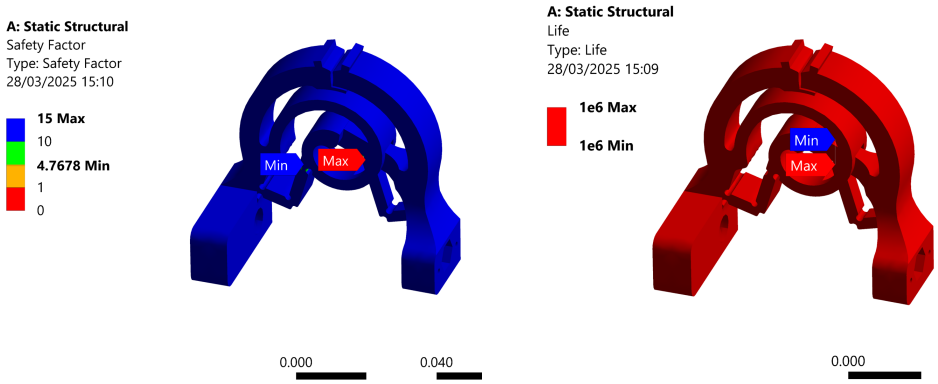


Figure 9: Fatigue SOF and Life

4. Hardware implementation

Lead Screw Mechanism:

A lead screw is a mechanical device that converts rotational motion into linear motion using a threaded shaft and a matching nut. When torque is applied to the screw, the nut moves linearly along the shaft. The output force generated by a lead screw is given by:

$$F_{\text{out}} = \frac{2\pi\eta T_{\text{in}}}{\text{pitch}}$$

where F_{out} is the output force, η is the efficiency, T_{in} is the input torque, and pitch is the distance between adjacent threads.

The linear displacement (stroke-length) for a given rotation angle θ is:

$$\text{Stroke-length} = \frac{\text{Pitch} \times \theta}{360}$$

For example, with a pitch of 1 mm and a rotation of 180°, the stroke-length is 0.5 mm.

Slider-Crank Mechanism:

A slider-crank mechanism converts rotational motion into linear motion using a crank arm, connecting rod, and slider. The output force for a given crank angle θ is:

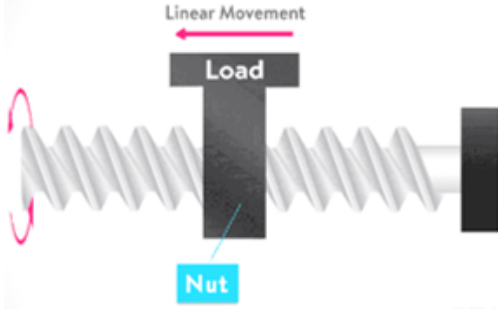
$$F_{\text{out}} = \frac{T_{\text{in}}}{OA \cdot \cos(\theta)}$$

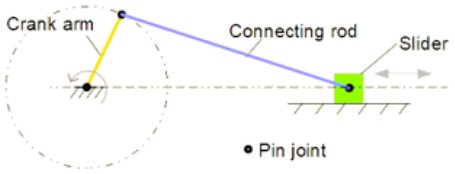
where OA is the crank radius. The output force varies with the crank angle, as shown in the adjacent graph.

Comparison:

- For the same input torque (0.2 Nm), a lead screw with a pitch of 1 mm and efficiency of 0.35 can generate a much higher output force (439.8 N) compared to a slider-crank mechanism with a crank radius of 8 mm (25 N).
- The lead screw provides precise and high-force linear motion, while the slider-crank is suitable for applications requiring continuous rotary-to-linear conversion with moderate force.

Mechanism comparisons



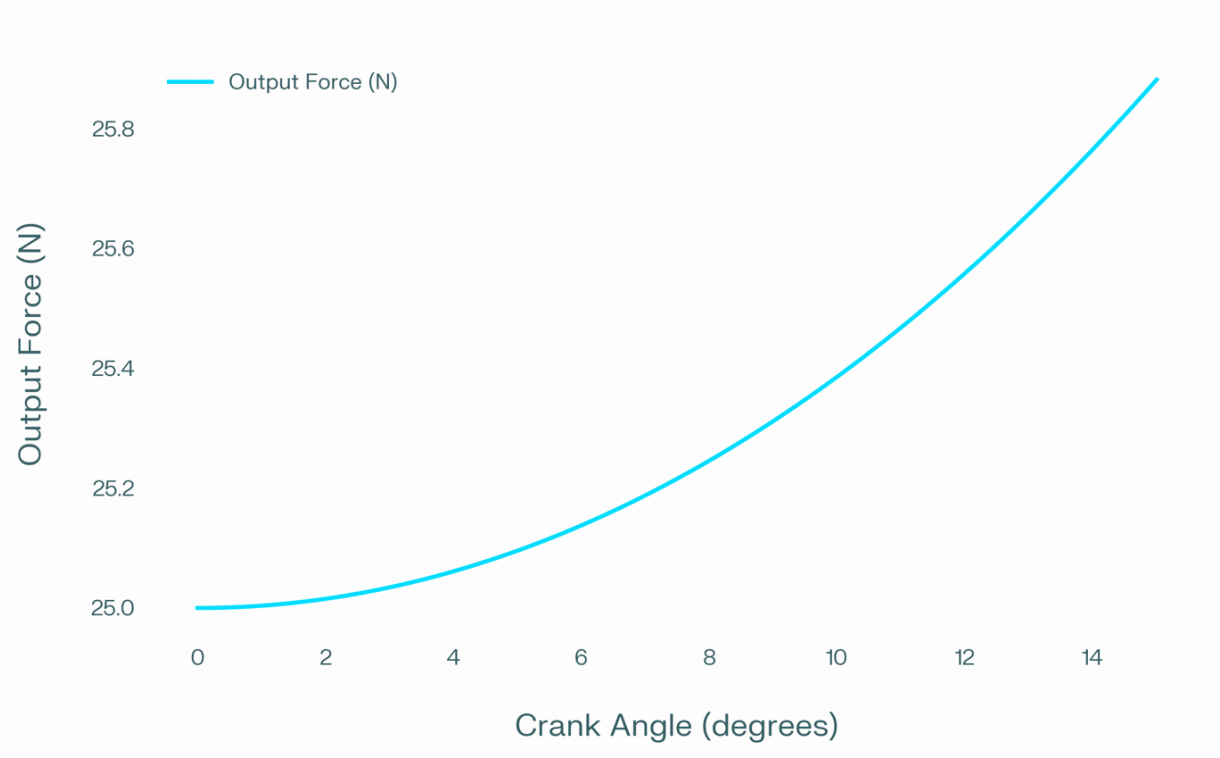


$$F_{\text{out}} = \frac{2\pi\eta T_{\text{in}}}{\text{pitch}}$$

$$\text{Stroke-length} = \frac{\text{Pitch} \times \theta}{360} = \frac{1 \times 180}{360} = 0.5\,\text{mm}$$

$$F_{\text{out}} = \frac{T_{\text{in}}}{OA \cdot \cos(\theta)}$$

Mechanism	Input Torque Nm	Key Parameters	Max Output Force N
Slider-Crank	0.2	Crank radius = 8 mm (0.008 m)	25
Lead Screw	0.2	Pitch = 1 mm (0.001 m), Efficiency = 0.35	439.8



4.A. Design Iterations

This slide illustrates the evolution of the mechanism design, highlighting the reasons for initial failure and the modifications that led to a successful outcome.

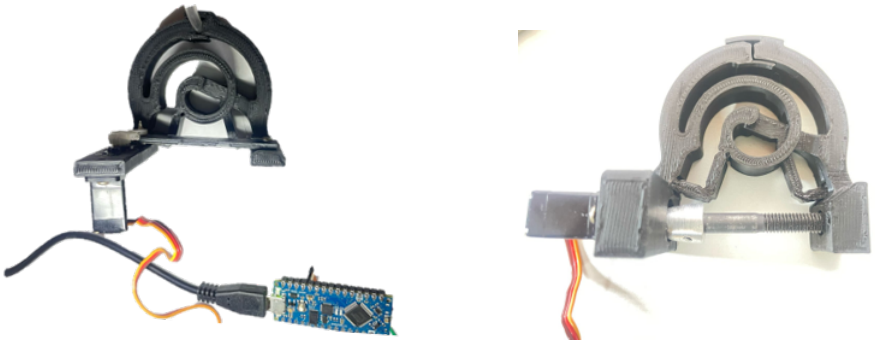


Figure 10: Fatigue SOF and Life

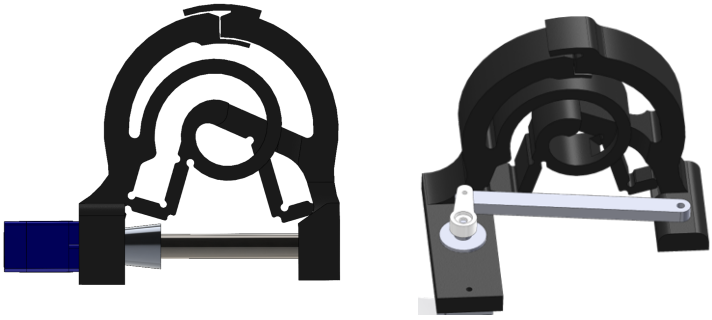


Figure 11: Fatigue SOF and Life

Failed Design:

- The initial design failed primarily because the stroke length was insufficient for the application's requirements.
- This limitation was due to the servo motor's restricted rotation of only 180°, which was not enough to produce the necessary linear displacement.
- Additionally, the model required a high output force, further challenging the mechanism's effectiveness.

The stroke length for the lead screw mechanism is given by:

$$\text{Stroke-length} = \frac{\text{Pitch} \times \theta}{360} = \frac{1 \times 180}{360} = 0.5 \text{ mm}$$

Worked Design:

- To address the failure, the design was modified to increase the stroke length, making it more suitable for the application's needs.
- However, this modification resulted in less generated force, and the servo torque was limited to 0.456 Nm.

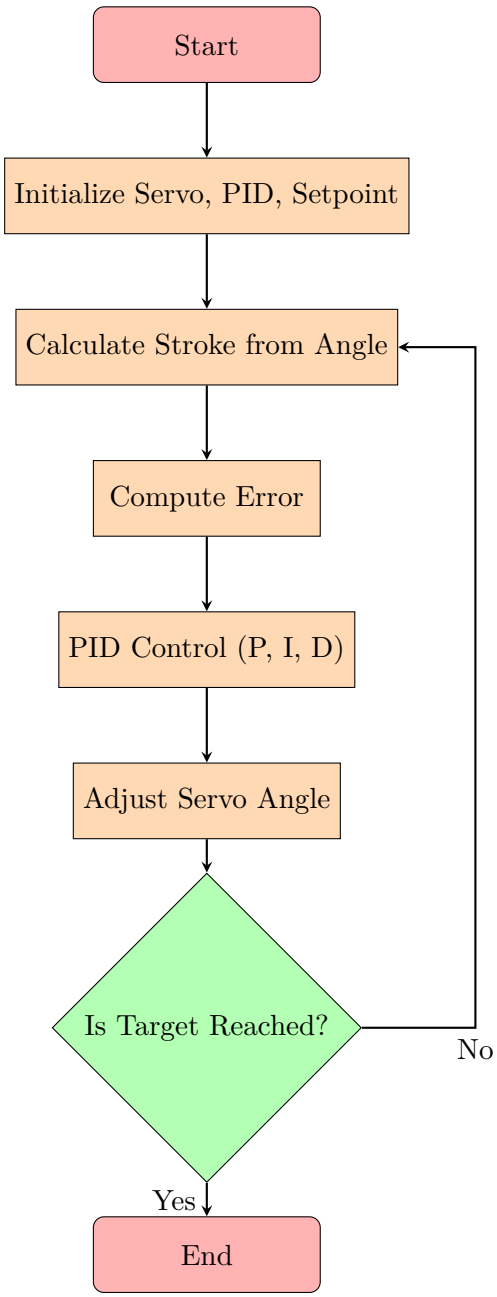
The output force for the slider-crank mechanism is calculated as:

$$F_{\text{out}} = \frac{T_{\text{in}}}{OA \cdot \cos(\theta)}$$

4.B. Controller

This pseudocode implements a real-time control system for a crank-slider mechanism actuated by a servo motor. The mechanism converts rotary motion into linear displacement, and the system aims to precisely control this stroke length using a PID controller. It begins by initializing the servo and setting control parameters, including PID gains and a target stroke value. The current stroke is calculated from the servo angle using trigonometric relationships based on crank-slider geometry.

A feedback loop then compares the current stroke with the desired setpoint and computes the error. The PID controller processes this error over time—accounting for its magnitude (proportional), accumulation (integral), and rate of change (derivative)—to generate a correction signal. This correction adjusts the servo angle, gradually driving the stroke toward the target position. The control loop runs continuously, and debug information is printed to monitor system performance. This approach allows for precise and adaptive control of non-linear mechanical systems in real time.



Stroke Control Loop for Crank-Slider Mechanism

Algorithm 1 Crank-Slider Stroke Control with PID and Servo

Initialize Servo on pin 9
Define crank radius r and connecting rod length l
Set PID constants: K_p, K_i, K_d
Set initial target stroke position (`setpoint`)
Initialize stroke position (`currentX`), servo angle (`angle`)
Initialize PID state: `error`, `prevError`, `integral`, `derivative`
Initialize previous time variable
Flag `initialized` $\leftarrow$ `false`
procedure SETUP
    Start serial communication
    Attach servo to pin
    Set servo to 0°
    Initialize `angle` and `prevTime`
procedure LOOP
    if not `initialized` **then**
        Move servo to 0°
        Wait for 1 second
        `initialized` $\leftarrow$ `true`
        return
    Convert `angle` to radians θ
    `sinPart` $\leftarrow r \cdot \sin(\theta)$
    `insideSqrt` $\leftarrow l^2 -$ `sinPart`^{2}
    if `insideSqrt` < 0 **then**
        `insideSqrt` $\leftarrow 0$
    `currentX` $\leftarrow r \cdot \cos(\theta) + \sqrt{\texttt{insideSqrt}}$
    `error` $\leftarrow$ `setpoint` $-$ `currentX`
    $dt \leftarrow$ `currentTime` $-$ `prevTime`
    `integral` $\leftarrow$ `integral` $+$ `error` $\cdot\, dt$
    `derivative` $\leftarrow$ (`error` $-$ `prevError`)$/dt$
    `output` $\leftarrow K_p \cdot$ `error` $+ K_i \cdot$ `integral` $+ K_d \cdot$ `derivative`
    `angle` $\leftarrow$ `angle` $-$ `output`
    Clamp `angle` between 0° and 180°
    Move servo to new `angle`
    Print debug info: target, current stroke, angle
    `prevError` $\leftarrow$ `error`
    `prevTime` $\leftarrow$ `currentTime`
    Wait 50 milliseconds

Pipe flow control is essential in industrial and process engineering to maintain desired flow rates and pressures. Controllers, such as PID (Proportional-Integral-Derivative), are widely used to regulate the system's output and ensure stability and fast response. This document presents a controller design for pipe flow control, simulation results, and a comparison of baseline and controlled systems.

5. Controller Design

The control system consists of a feedback loop with a PID controller, a nonlinear force model, and a first-order plant. The block diagram of the control system is shown in Figure 12.

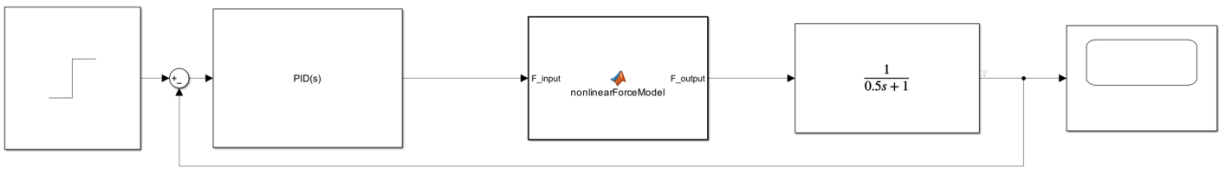


Figure 12: Block diagram of the pipe flow control system. The system includes a PID controller, a nonlinear force model, and a first-order plant.

The controller receives the error between the desired and actual output, processes it through the PID block, and applies corrective force to the nonlinear plant. The plant dynamics are modeled as a first-order system with transfer function $\frac{1}{0.5s+1}$.

5.A. Baseline Model (No Controller)

The step response of the baseline model (without controller) is shown in Figure 13. The system output rises gradually and settles at the desired value, but with a slower response and potential steady-state error.

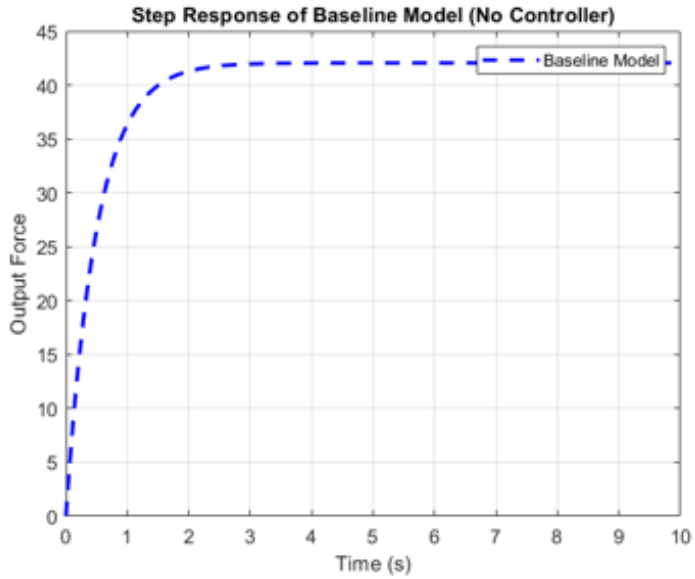


Figure 13: Step response of the baseline model without controller. The system shows a slower rise and takes longer to reach steady state.

5.B. Controlled System

Figure 14 presents the step response of the controlled system. With the PID controller, the output force rapidly reaches the setpoint with minimal overshoot and steady-state error, demonstrating improved performance.

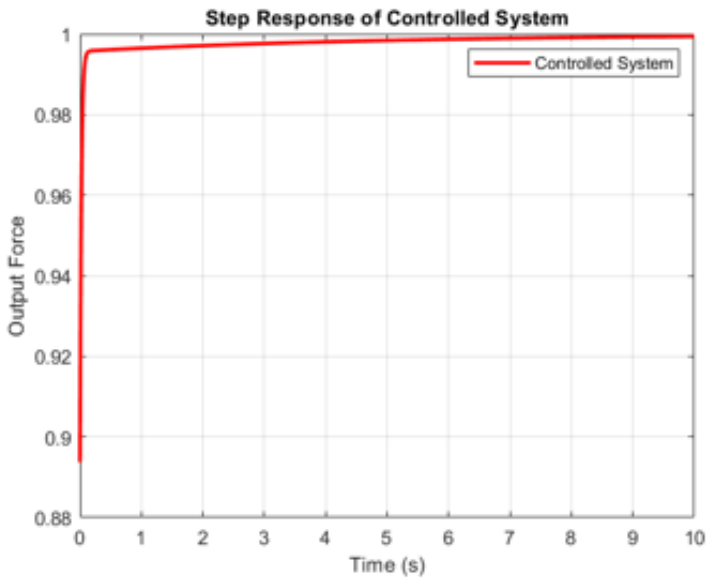


Figure 14: Step response of the controlled system. The controller ensures a fast, stable response with negligible steady-state error.

5.C. Comparison of Baseline and Controlled Systems

A direct comparison of the baseline and controlled system responses is illustrated in Figure 15. The controlled system (red) clearly outperforms the baseline (blue), achieving faster rise time and better tracking of the desired output.

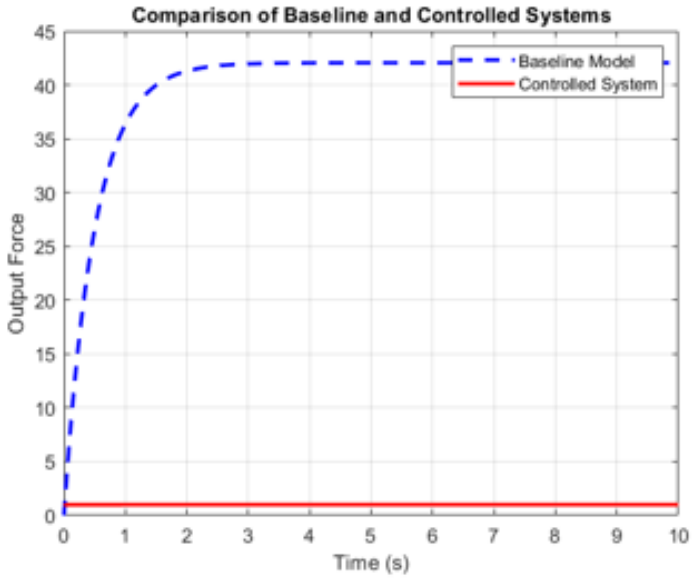


Figure 15: Comparison of baseline and controlled system responses. The controlled system achieves superior performance in terms of speed and accuracy.

The designed PID controller significantly enhances the pipe flow system's performance, as evidenced by the simulation results. The controller ensures quick settling, minimal overshoot, and negligible steady-state error compared to the uncontrolled baseline system.

6. Conclusion

This work presents a novel spiral spring-based compliant microgripper design with integrated pipe flow control, validated through both simulation and experimental results. The design approach leverages a PID-controlled crank-slider mechanism (see Figures 12–15) to achieve precise and adaptive force-displacement characteristics, essential for applications such as medical drug delivery systems.

The novelty of this work lies in the use of a spiral spring as a compliant, energy-storing element, which is not found in prior literature for this application. This approach enables a higher lifecycle, enhanced miniaturization, and simplicity compared to traditional actuation methods (e.g., SMA wires). The design achieves a 55-fold stiffness variation per hinge, allowing for adaptive grasping and precise flow regulation, outperforming existing methods in both flexibility and reliability.

Design efficiency is demonstrated by the compact, lightweight 3D-printed PLA+ structure (46×46×11 mm, 3 g), which achieves a miniaturizing ratio of 0.5 and can handle objects up to 4 mm. Kinematic and force-deflection analyses confirm a balanced relationship between input force and output displacement, critical for precise flow control.

Finite Element Analysis (FEA) results validate the structural robustness, with maximum deformation limited to 3.36 mm under operational loads and a Young's modulus of 2.06–3.64 GPa, confirming the suitability of PLA+ for high-cycle fatigue applications. Comparative analysis shows that the lead screw actuation mechanism provides higher output force compared to slider-crank mechanisms, though with a trade-off in stroke length.

The experimental hardware implementation, including the PID-based control algorithm, demonstrates the practical feasibility and effectiveness of the system in real-world pipe flow control scenarios. The controller design and performance are illustrated in Figures 12–15: the controlled system exhibits faster settling, minimal overshoot, and negligible steady-state error compared to the baseline.

No prior studies on this specific spiral spring-based compliant mechanism for microgrippers were found, underscoring the originality and impact of this research. Future work should focus on optimizing geometry and material selection at critical joints to further improve fatigue life and conducting additional experimental validation under real-world conditions.

References

[1] Baek H. Shin B. Kim Y. Ali H.F.M., Khan A.M. Modeling and control of a finger-like mechanism using bending shape memory alloys. *Microsystem Technologies*, 27:2481–2492, 2021.

[2] Jala V.R. Ashrafiuon H. Sliding mode control of mechanical systems actuated by shape memory alloy. *Journal of Dynamic Systems Measurement Control*, 131:1–6, 2009.

[3] Libu George Babu and Bharanidaran Ramalingam. Design of compliant gripper for surgical applications. *Australian Journal of Mechanical Engineering*, 2019.

[4] Kyler C. Bingham, Matthew Hessler, Safal Lama, and Taher Deemyad. Design and implementation of a compliant gripper for form closure of diverse objects. *Applied Sciences*, 13:9677, 2023.

[5] D. Farhadi Machekposhti, N. Tolou, and J. L. Herder. A review on compliant joints and rigid-body constant velocity universal joints toward the design of compliant homokinetic couplings. *Journal of Mechanical Design*, 137(3):032301, 03 2015.

[6] G. Hao and J. Zhu. Design of a monolithic double-slider based compliant gripper with large displacement and anti-buckling ability. *Micromachines*, 10(10):665, 2019.

[7] Ananthasuresh G.K. Kumar R.G. A study of mechanical advantage in compliant mechanisms. In *Proceedings of the First International Conference on Machines & Mechanisms (iNaCoMM)*, pages 18–20 December, 2013.

[8] Mariangela Manti, Taimoor Hassan, Giovanni Passetti, Nicolò D'Elia, Cecilia Laschi, and Matteo Cianchetti. A bioinspired soft robotic gripper for adaptable and effective grasping. *Soft Robotics*, 2(3):107–116, 2015.

[9] Phan T.V. Nguyen, D.C. and H.T. Pham. Design and analysis of a compliant gripper integrated with constant-force and static balanced mechanism for micromanipulation. In *2018 International Conference on Green Technology and Sustainable Development (GTSD)*, pages 291–295, 2018.

[10] Fleck N.A. Ananthasuresh G.K. Pedersen, C.B.W. Design of a compliant mechanism to modify an actuator characteristic to deliver a constant output force. *Journal of Mechanical Design*, 128:1101–1112, 2006.

[11] Dhanalakshmi K. Choi S.B. Ruth D.J.S., Sohn J.W. Control aspects of shape memory alloys in robotics applications: A review over the last decade. *Sensors*, 22:4860, 2022.

[12] Ganapathy Then Mozhi, Kaliaperumal Dhanalakshmi, and Seung-Bok Choi. Design and control of monolithic compliant gripper using shape memory alloy wires. *Sensors*, 23:2052, 2023.

[13] Lin H.-T. Baryshyan A. Leisk G. Kaplan D. Trimmer, B.A. Towards a biomorphic soft robot: Design constraints and solutions. In *Proceedings of the IEEE RAS and EMBS International Conference on Biomedical Robotics and Biomechatronics*, pages 599–605, 2012.

[14] Q. Xu. Structure design of a new compliant gripper based on scott-russell mechanism. In *2013 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1623–1628, 2013.

Appendix

A. Structure of Code

Force deflection analytical model code

```
import math
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

class Kinematics:
    "forward and inverse kinematics calc"

    def __init__(self, L):
        self.L = L  # Length parameter

    "x from theta x = L sin(theta)"
    def forward_kinematics(self, theta):

        return self.L * math.sin(theta)

    "theta from x using theta = x / L"
    def inverse_kinematics(self, x):

        return x / self.L
class ForceDeflection:
    "force-deflection relation"

    def __init__(self, ke, E, I, L):
        self.ke = ke  # stiffness coeff
        self.E = E    # Youngs modulus
        self.I = I    # moment of inertia
        self.L = L    # Length

    "Computeing x = (Fin * L^3) / (ke * E * I)"
    def displacement(self, Fin):
        return (Fin * self.L3) / (self.ke * self.E * self.I)


    "Computes Fin =(ke * E * I / L^3) * x"
    def required_input_force(self, x):
        return (self.ke * self.E * self.I / self.L3) * x
class MomentBalance:
    "moment balance at pivot"

    def __init__(self, k, Lc, Lh, Fin, Lin, Le, theta_values):
        """ k: List of stiffness coeff k0, k1, k2, ...
            Lc, Lh: Length parameters
            Fin, Lin
            Le
            theta_values: List of thetaa 0,1,2,3,4,5,6  """
        self.k = k
        self.Lc = Lc
        self.Lh = Lh
        self.Fin = Fin
        self.Lin = Lin
        self.Le = Le
        self.theta_values = theta_values
##
##
##
```

```
    def compute_Fout(self):
        "output fource Fout"
        numerator = sum(self.k[i] * self.theta_values[i] for i in range(len(self
                                                .k))) + self.Fin * self.Lin ###
                                                consider -ve values for k4, k5,
                                                k6 while giving values to k
        return numerator / self.Le

class ForceTransformation:
    "Handles force transformation equations"

    def __init__(self, Lc, Lh, theta_h, theta_c):
        self.Lc = Lc  # Length (c)
        self.Lh = Lh  # Length (h)
        self.theta_h = theta_h
        self.theta_c = theta_c

    "Computes transformed forces"
    def compute_forces(self, Xc, Xh, Fc):

        Xh_new = Xc * (self.Lh / self.Lc) * (math.sin(self.theta_h) / math.sin(
                                                self.theta_c))
        Fh = Fc * (self.Lc / self.Lh) * (math.sin(self.theta_c) / math.sin(self.
                                                theta_h))
        return Fh, Xh_new

    "Computes output force using Fc * Xc = Fh * Xh"
    def compute_output_force(self, Fc, Xc, Xh):
        return (Fc * Xc) / Xh
if __name__ == "__main__":
    # Define constants
    L = 10
    ke, E, I = 1.5, 200, 0.005                                          ####
                                            stiffness coeff, Youngs modulus,
                                            moment if inertia
    Lc, Lh = 5, 7                                                       ####
    Fin, Lin, Le = 50, 2, 3
    theta_values = [0.1, 0.2, 0.15, 0.3, 0.25, 0.35]
    k = [10, 12, 14, 16, 18, 20]

    # Kinematics
    kin = Kinematics(L)
    theta = 0.1
    x = kin.forward_kinematics(theta)
    print(f"Forward Kinematics: x = {x}\n")

    # Force Deflection
    fd = ForceDeflection(ke, E, I, L)
    Fin_required = fd.required_input_force(x)
    print(f"Required Input Force: Fin = {Fin_required}\n")

    # Moment Balance
    mb = MomentBalance(k, Lc, Lh, Fin, Lin, Le, theta_values)
    Fout = mb.compute_Fout()
    print(f"Computed Output Force: Fout = {Fout}\n")

    # Force Transformation
    ft = ForceTransformation(Lc, Lh, math.radians(30), math.radians(45))
    Xc, Xh, Fc = 2.5, 3.0, 100
    Fh, Xh_new = ft.compute_forces(Xc, Xh, Fc)
    print(f"Transformed Forces: Fh = {Fh},\tXh_new = {Xh_new}\n")
```

```
    Fc_out = ft.compute_output_force(Fc, Xc, Xh_new)
    print(f"Output Force: Fc_out = {Fc_out}\n")


Fin_values = np.linspace(0, 1.9, 50)  # Vary Fin from 0 to 100 with 50 points
Fout_values = []

# Compute Fout for each Fin
for Fin in Fin_values:
    moment_balance = MomentBalance(k, Lc, Lh, Fin, Lin, Le, theta_values)
    Fout_values.append(moment_balance.compute_Fout())

# Plot the graph
plt.figure(figsize=(8, 8))
plt.plot(Fin_values, Fout_values, marker='o', linestyle='-', color='b', label='
                                        Fout vs Fin')

plt.xlabel('Input Force (Fin)')
plt.ylabel('Output Force (Fout)')
plt.title('Relationship between Fin and Fout')
plt.legend()
plt.grid()
plt.show()



deflection_values = []
force_deflection = ForceDeflection(ke, E, I, L)

for Fin in Fin_values:
    # Compute deflection
    deflection_values.append(force_deflection.displacement(Fin))



# Plot Fin vs Deflection
plt.subplot(1, 2, 2)
plt.plot(Fin_values, deflection_values, marker='s', linestyle='-', color='r',
                                        label='Deflection vs Fin')
plt.xlabel('Input Force (Fin)')
plt.ylabel('Deflection (x)')
plt.title('Force-Deflection Relationship')
plt.legend()
plt.grid()

plt.tight_layout()
plt.show()
```

```
#include <Servo.h>

Servo myServo;

// Crank-slider mechanism constants
const float r = 9.0;   // Crank radius in mm
const float l = 56.0;  // Connecting rod length in mm

// PID constants
float Kp = 3.0;
float Ki = 0.1;
float Kd = 0.5;

// Stroke control variables
float setpoint = 5.0;    // Target stroke in mm
```

```
float currentX = 0.0;     // Calculated stroke in mm
float angle = 0.0;        // Servo angle in degrees

// PID state
float error = 0.0, prevError = 0.0, integral = 0.0, derivative = 0.0;
unsigned long prevTime = 0;

bool initialized = false;  // To ensure it first goes to 0

void setup() {
  Serial.begin(9600);
  myServo.attach(9);
  myServo.write(0);  // Force initial position to 0
  angle = 0.0;
  prevTime = millis();
}

void loop() {
  // 1. Move to initial 0   position just once
  if (!initialized) {
    angle = 0;
    myServo.write(angle);
    delay(1000);  // Give it time to settle
    initialized = true;
    return;
  }

  // 2. Calculate crank-slider position from angle
  float thetaRad = radians(angle);
  float sinPart = r * sin(thetaRad);
  float insideSqrt = sq(l) - sq(sinPart);
  if (insideSqrt < 0) insideSqrt = 0;

  currentX = r * cos(thetaRad) + sqrt(insideSqrt);  // in mm

  // 3. PID control
  error = setpoint - currentX;
  unsigned long now = millis();
  float dt = (now - prevTime) / 1000.0;

  integral += error * dt;
  derivative = (error - prevError) / dt;

  float output = Kp * error + Ki * integral + Kd * derivative;

  angle -= output;
  angle = constrain(angle, 0, 180);  // Ensure valid servo range
  myServo.write(angle);

  // 4. Debug output
  Serial.print("Target: ");
  Serial.print(setpoint);
  Serial.print(" mm | Current: ");
  Serial.print(currentX);
  Serial.print(" mm | Angle: ");
  Serial.println(angle);

  prevError = error;
  prevTime = now;

  delay(50);
}
```