ME340 Lab Report 3

Shihong Yuan

Lab TA: Tim Han
Lab Session: Friday 1:30am-2:50am
October 17, 2025

Inlab Q1: Link Length Calculation

Code

```
link_lengths=np.zeros((len(connectivity_matrix),)) #Allocate array for link lengths
for i, link in enumerate(connectivity_matrix):
     #loop over connectivity matrix
     node1_pos=initial_node_positions[link[0]]
     node2_pos=initial_node_positions[link[1]]
     link_lengths[i]=np.linalg.norm(node1_pos - node2_pos)
```

Listing 1: Python code for calculating link lengths.

Inlab Q2: Acceleration Calculation and Plot

Code

```
# Add Code to find the crank angle
crank_tip_node_id = set_diff(connectivity_matrix[crank_link_idx], motor_node_idx)

initial_crank_vector = initial_node_positions[crank_tip_node_id] -
    initial_node_positions[motor_node_idx]

initial_crank_angle_rad = np.arctan2(np.imag(initial_crank_vector), np.real(
    initial_crank_vector))
initial_crank_angle_deg = np.rad2deg(initial_crank_angle_rad)

crank_angle = initial_crank_angle_deg + crank_angular_velocity * time

# Add Plots as a function of crank angle
fig, ax1 = plt.subplots()
color = 'tab:blue'
ax1.set_xlabel('Crank Angle (degrees)')
ax1.set_ylabel('Velocity (m/s)', color=color)
ax1.plot(crank_angle, np.real(velocity[plotting_node_idx,:]), color=color, label='X-
    Velocity')
ax1.tick_params(axis='y', labelcolor=color)
ax1.grid(True)

ax2 = ax1.twinx()
color = 'tab:red'
ax2.set_ylabel('Acceleration (m/s^2)', color=color)
ax2.plot(crank_angle, np.real(acceleration[plotting_node_idx,:]), color=color, label=
    'X-Acceleration')
ax2.tick_params(axis='y', labelcolor=color)

lines, labels = ax1.get_legend_handles_labels()
lines2, labels2 = ax2.get_legend_handles_labels()
ax2.legend(lines + lines2, labels + labels2, loc='upper right')

plt.title('Output Rocker Node X-Velocity and X-Acceleration vs. Crank Angle')

fig.tight_layout()
plt.show()
plt.close()
```

Listing 2: Python code for Acceleration Calculation and Plot.

Inlab Q3: Crank-Rocker Mechanism Simulation

```python
initial_node_positions = np.array([0+0j,1+5j,2+1j,3]) # Node/Joint Placement
connectivity_matrix = [[0,1],[1,2],[2,3],[3,0]] #Connectivity Matrix
ground_links_idx = [3] #indices of grounded links (index of row in connectivity
    matrix)
crank_link_idx = 2 # Link with prescribed position over time (input crank)
motor_node_idx = 3 #Node the motor is at.
sliders = [] #Slider constraints #[node index,link 1 index, link 2 index] or [node
    index,link index, x direction, y direction]
links_with_fixed_angle = [] #Fix link/link angle during rotation. (Must share a node)
rotation_fixed_nodes = [] #Nodes that are fixed so that they cant rotate
slider_friction = [] #Friction at slider constraints

tperiod= 36 #total time to run simulation (seconds). Try to keep this small to
    minimize computational time.
dt=0.05 #time step (seconds). If code takes too long to converge, increase dt.
crank_angular_velocity= -10#Angular velocity of input crank (deg/sec) -Assumed
    constant.
              #+(-) angular velocity denotes CCW (CW) rotation
plotting_node_idx = 1 # for plotting the position, velocity, acceleration over time
```

Listing 3: Python code for Crank-Rocker Mechanism Simulation

Simulation Result

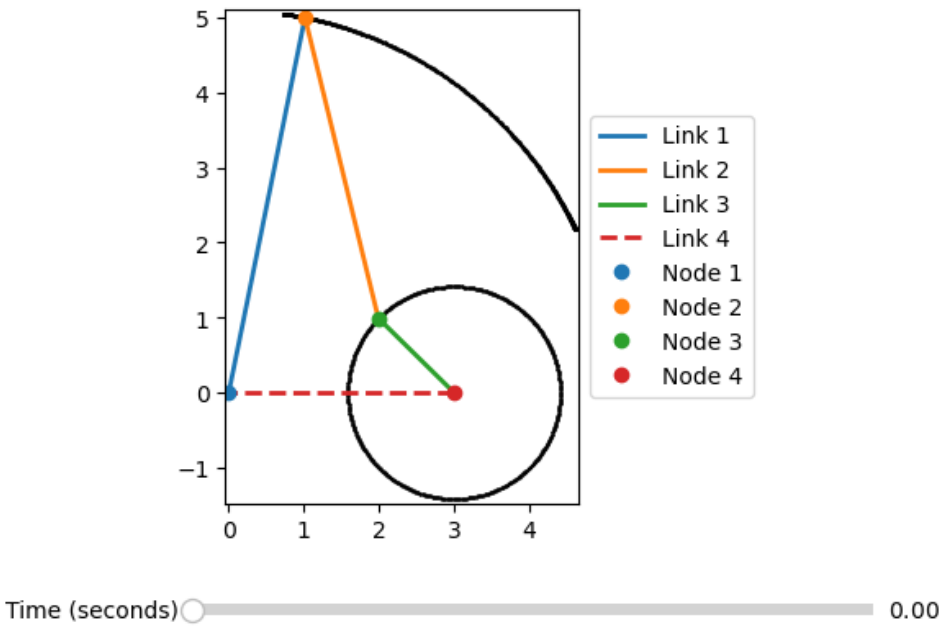

Figure 1: Output1

AI Usage Statement

The following option was selected concerning the use of AI in this assignment:

a) My answer was created by a Gen AI algorithm, and I have not modified it

b) My answer was created by a Gen AI algorithm, and I have made some minor changes.

c) My answer was created by a Gen AI algorithm, and I have made major changes.

d) My answer was created solely by myself.

e) If I used Gen AI, I used ____ (name of program).

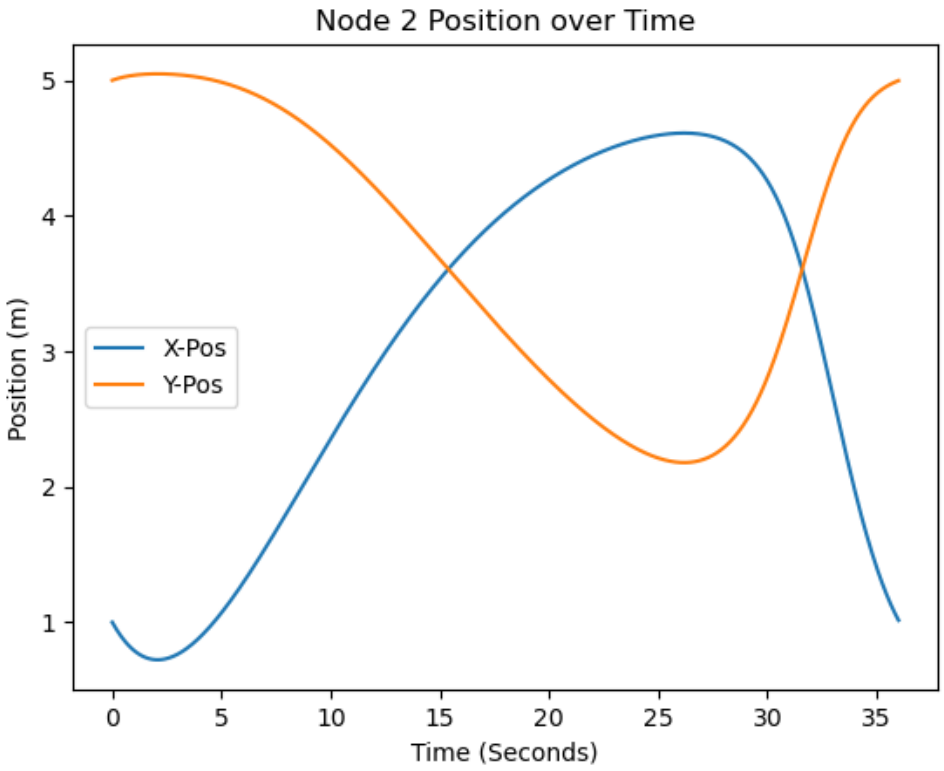

Figure 2: Output2

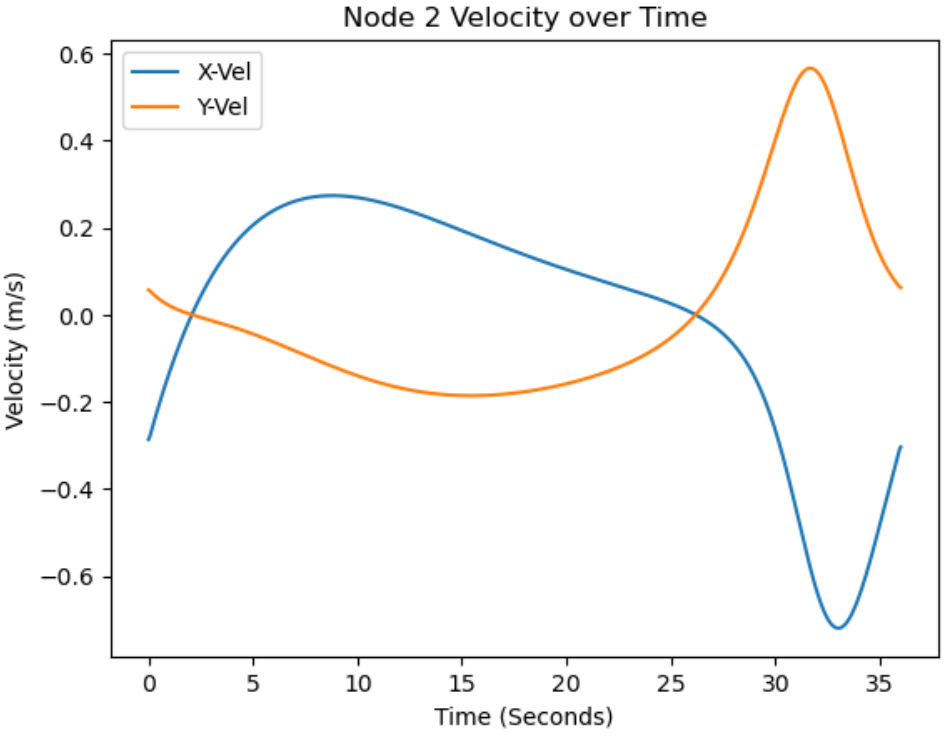

Figure 3: Output3

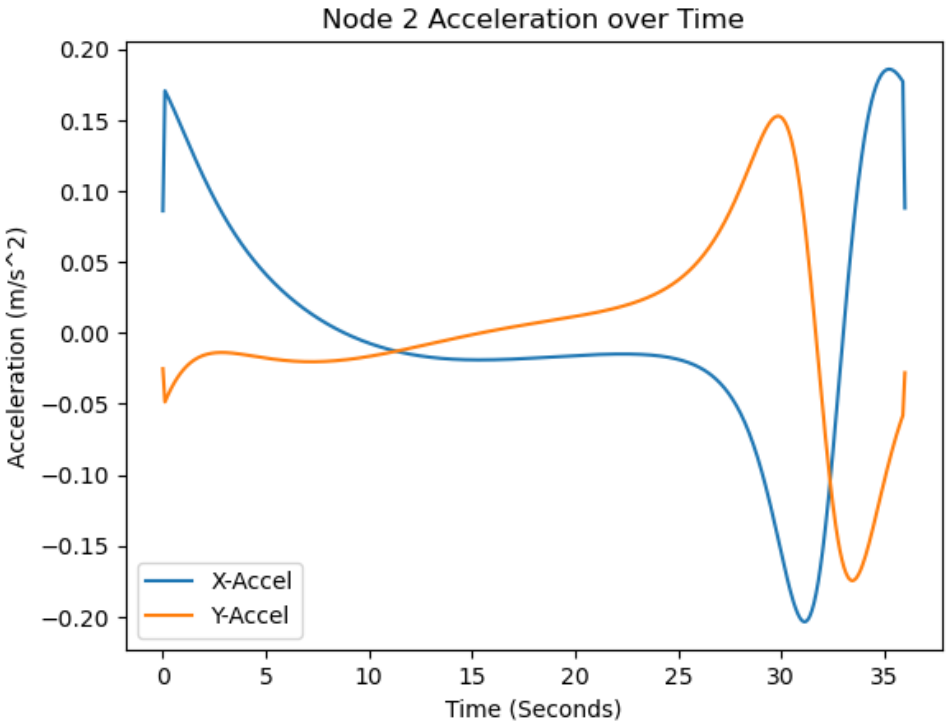

Figure 4: Output4

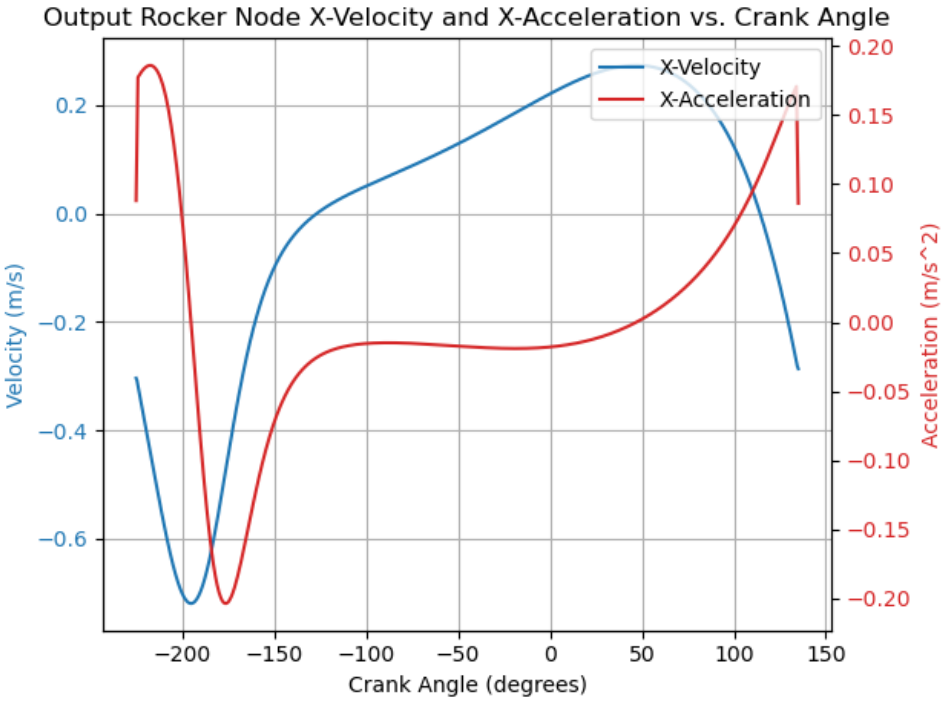

Figure 5: Output5

Postlab Q1: Plots of Slider Motion

The following plots show the position, velocity, and acceleration of the horizontal slider (node 5) as a function of time for four full cycles of the crank. The crank angular velocity was set to 40 degrees/second to complete one revolution in 9 seconds.

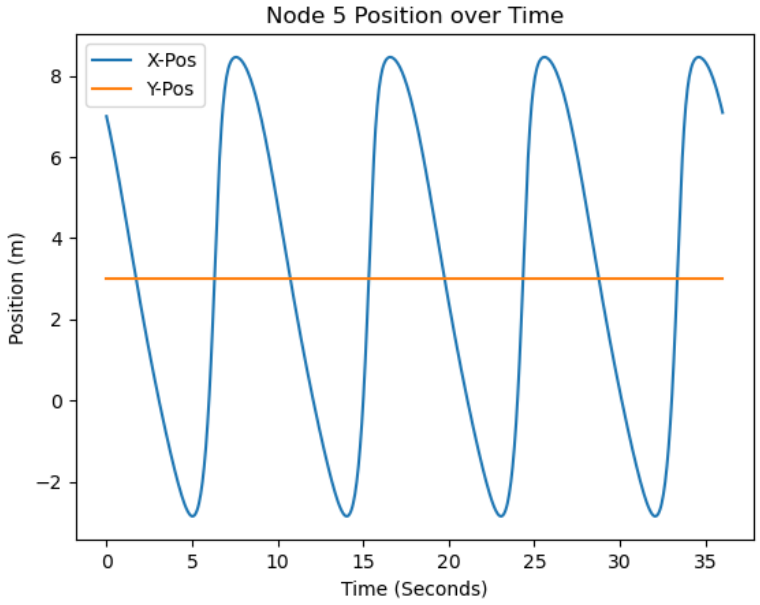

Figure 6: Position of the horizontal slider (Node 5) vs. Time.

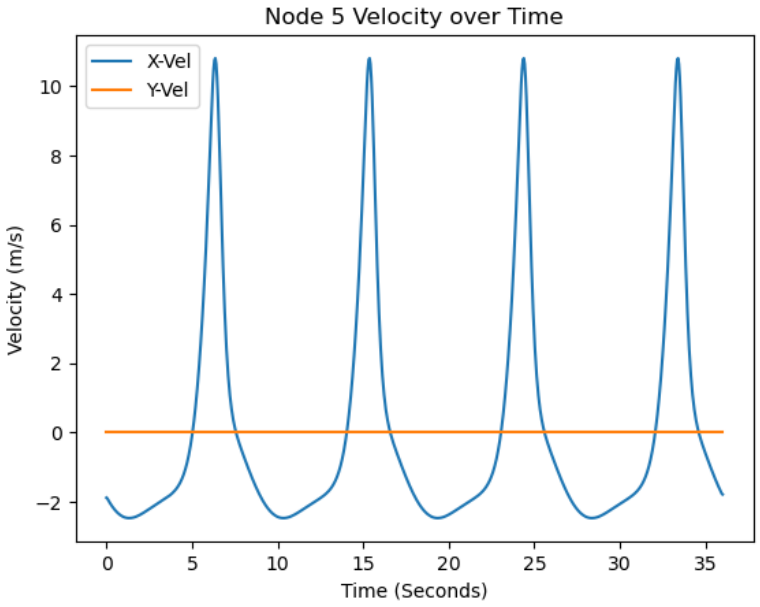

Figure 7: Velocity of the horizontal slider (Node 5) vs. Time.

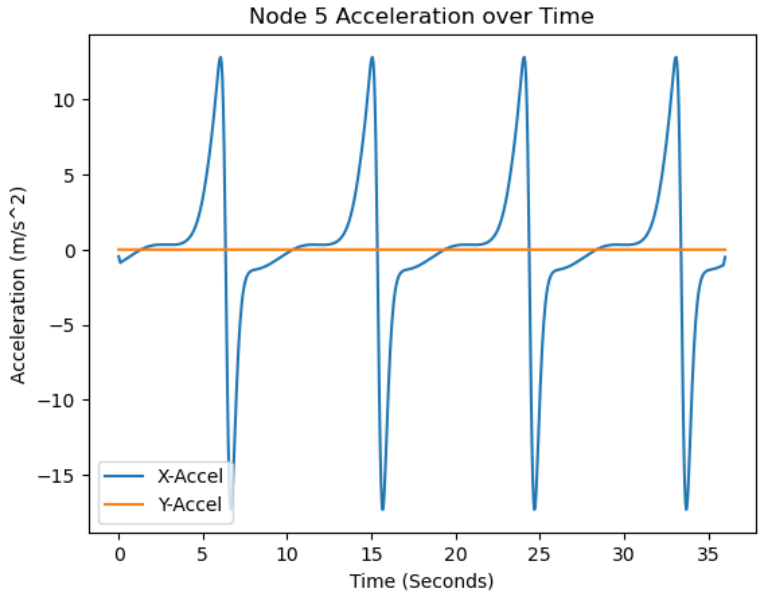

Figure 8: Acceleration of the horizontal slider (Node 5) vs. Time.

Postlab Q2: Modified Code and Commentary

Code Used for Simulation

The following Python code block was configured to define the geometry, constraints, and motion parameters for the Whitmore quick return mechanism.

```
# Node positions are 0-indexed (Node 1=0, etc.). Values from the plot in Fig 1.

initial_node_positions = np.array([0+0j, 0-5j, 3+1j, 4+3j, 7+3j])

# Link 1:[0,2], Link 2:[0,1], Link 3:[1,2], Link 4:[2,3], Link 5:[3,4], Link 6:[1,4]
connectivity_matrix = [[0, 2], [0, 1], [1, 2], [2, 3], [3, 4], [1, 4]]

# Links at indices 1 and 5 are fixed to ground.
ground_links_idx = [1, 5]
crank_link_idx = 0
motor_node_idx = 0

# Defines the two slider constraints in the system.
sliders = [[4, 1, 1, 0], [2, 2, 3]]
links_with_fixed_angle = [[2, 3]]
rotation_fixed_nodes = []
slider_friction = [0, 0]
tperiod= 36
dt=0.05

crank_angular_velocity= 40
plotting_node_idx = 4
```

Listing 4: Python code for Section 1 Input.

Explanation of Modifications

To model this specific linkage, several key parameters in Section 1 were defined based on the mechanism's geometry and required motion. The following is a comment on these modifications.

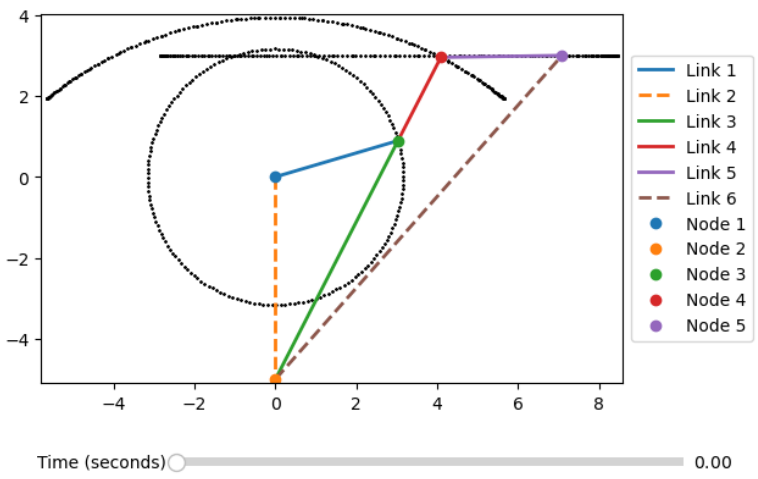

Figure 9: Simulation

1. Node Positions

The `initial_node_positions` array sets the starting coordinates for the 5 joints (nodes) of the mechanism in the 2D plane. Each position is represented as a complex number $(x + yj)$. For this configuration, the nodes were initialized at:

- **Node 1:** $(0, 0)$
- **Node 2:** $(0, -5)$
- **Node 3:** $(3, 1)$
- **Node 4:** $(4, 3)$
- **Node 5:** $(7, 3)$

2. Link Connectivity

The `connectivity_matrix` defines the 6 links that form the structure of the mechanism. Each entry `[node_A, node_B]` represents a rigid link connecting two nodes. This specific matrix builds the required linkage, including the crank, the ground frame, and the slotted arm.

3. Ground Links

The `ground_links_idx` array specifies which of the links in the `connectivity_matrix` are fixed and do not move. Here, the links at **index 1** (connecting Node 1 and Node 2) and **index 5** (connecting Node 2 and Node 5) are designated as the stationary ground frame of the mechanism.

4. Slider Constraints

Two types of sliding joints were modeled using the `sliders` and `links_with_fixed_angle` variables.

- **Slider 1 (Fixed Track Slider):** This constraint, `[4, 1, 1, 0]`, models a **fixed slider**. It dictates that **Node 5** (the 4th index) is constrained to move along a path relative to **Link 2** (the 1st index). Since Link 2 is a ground link, this means Node 5 slides along a fixed track. The direction vector `[1, 0]` specifies that this motion is purely **horizontal**.

- **Slider 2 (Moving Track Slider):** The second constraint models **Node 3** (index 2) sliding along the moving, slotted arm. This was achieved by defining the slotted arm as two separate but collinear links in the `connectivity_matrix` (the links at index 2 and 3).
  - The slider entry `[2, 2, 3]` constrains Node 3 to only move along the line formed by these two links.
  - To ensure these two links act as a single rigid body, the `links_with_fixed_angle = [[2, 3]]` command was added. This forces the angle between them to remain constant, keeping them perfectly aligned throughout the simulation.

Postelab Q3. AI Statement

Select one of the following options:
a) My answer was created by a Gen AI algorithm, and I have not modified it
b) My answer was created by a Gen AI algorithm, and I have some minor changes.
c) My answer was created by a Gen AI algorithm, and I have made major changes.
d) My answer was created solely by myself.