Week HW 1, KIN PART ROT PLANE1

Anas Khmais Hamrouni

January 29, 2025

1 Task 1:

You should find:

1. simulate the move of $\vec{O}$ for $t = [0..10]$;
2. find and draw plots v, a, a_n, a_τ, κ (Osculating circle) respect to t;
3. find $y(x)$, $\vec{v}$, $\vec{a}$, $\vec{a}_n$, $\vec{a}_\tau$ and show it on the simulation.

$$\vec{O} = \begin{cases} x = 3\cos(2t)\cos(t) + 0.82 \\ y = 3\cos(2t)\sin(t) + 0.82 \end{cases}$$

1.1 Overview

This program simulates a parametric trajectory defined by the equations:

$$x(t) = 3\cos(2t)\cos(t) + 0.82, \quad y(t) = 3\cos(2t)\sin(t) + 0.82.$$

The simulation computes velocity, acceleration, tangential acceleration, normal acceleration, and curvature as functions of time.

1.2 Mathematical Background

The trajectory is governed by parametric equations for $x(t)$ and $y(t)$. From these, the velocity and acceleration components are derived:

$$v_x(t) = \frac{dx}{dt}, \quad v_y(t) = \frac{dy}{dt}, \quad a_x(t) = \frac{d^2x}{dt^2}, \quad a_y(t) = \frac{d^2y}{dt^2}.$$

The total velocity $v(t)$ and acceleration $a(t)$ are given by:

$$v(t) = \sqrt{v_x^2 + v_y^2}, \quad a(t) = \sqrt{a_x^2 + a_y^2}.$$

Tangential and normal accelerations are computed as:

$$a_\tau(t) = \frac{v_x a_x + v_y a_y}{v}, \quad a_n(t) = \sqrt{a^2 - a_\tau^2}.$$

The curvature $\kappa(t)$ is calculated as:

$$\kappa(t) = \frac{|v_x a_y - v_y a_x|}{v^3}.$$

1.3 Animation Description

The animation visualizes the trajectory and the vectors representing velocity, acceleration, tangential acceleration, and normal acceleration. It is created using the `matplotlib.animation` library. Each frame updates the position and vectors to match the current time step. Additionally, a second plot displays the time evolution of velocity, acceleration, tangential and normal accelerations, and curvature.

1.4 Code

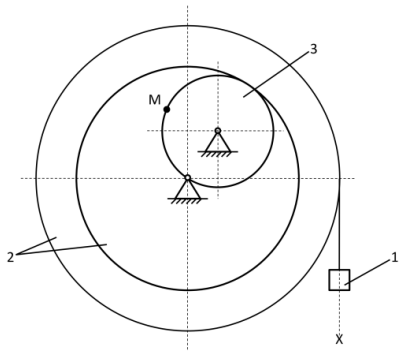
The full code solution is available on GitHub: GitHub Repository Link.

2 Task 2:

You should solve the task, till the M point travels s:

1. simulate this mechanism (obtain all positions of bodies 1, 2, 3)
2. velocity for M(draw plots for magnitudes and show vectors on simulation);
3. accelerations (tangent, normal, overall) for M(draw plots for magnitudes and show vectors on simulation);
4. draw plots of angular velocities for 2, 3 bodies.

If $R_2 = 40,\ r_2 = 30,\ R_3 = 15$
$x = x(t) = 3 + 80t^2,\ s_M = 4.$



Task 2
(Yablonskii (eng) K2

2.1 Mathematical Model

The motion of the system consists of three bodies: Body 1, Body 2 , and Body 3. We focus on calculating the kinematics of these bodies, specifically

the position, angular velocity, and acceleration of Body 1 and Body 3, and the velocity and acceleration of a point M on Body 3.

1. **Time Calculation:** The maximum time $t_{\max}$ is determined based on the arc length s_M of point M and the radius R_3 of Body 3:

$$t_{\max} = \sqrt{\frac{s_M}{4R_3}}$$

This formula calculates the total time of motion for the system, ensuring that the movement of the bodies is physically consistent with the given arc length.

2. **Position and Angle Calculations:** - The position of Body 1, x_{body1}, is given by the equation:

$$x_{\text{body1}} = 3 + 80t^2$$

where t is the time.

- The angle θ_2 of Body 2 is given by:

$$\theta_2 = 2t^2$$

and the angle θ_3 of Body 3 is:

$$\theta_3 = 4t^2$$

3. **Angular Velocities:** The angular velocities of Bodies 2 and 3 are computed as:

$$\omega_2 = 4t$$

$$\omega_3 = 8t$$

4. **Velocity and Acceleration of Point M:** The velocity of point M, v_M, is calculated as:

$$v_M = 120t$$

The tangential and normal accelerations of point M are given by:

$$a_t = 120$$

$$a_n = 960t^2$$

2.2 Animation Creation

The animation was implemented using Python's `matplotlib` and `matplotlib.animation` modules. The following steps outline the process:

1. **Initialization:** The system is initialized with default values for the arc length s_M, which is used to calculate the kinematic parameters such as position, velocity, and acceleration of the bodies.

3. **Dynamic Update of Components:** - The position of Body 1 is updated at each frame using the equation $y_{\text{body1}} = -x_{\text{body1}} - 20$, ensuring that the body moves as time progresses. - The position of point M on Body 3 is updated using polar coordinates, based on the angle θ_3:

$$x_M = R_3 \cos(\theta_3), \quad y_M = R_3 \sin(\theta_3)$$

The velocity and acceleration vectors at point M are also updated at each frame. These vectors are displayed using `quiver` plots for visualization.

This animation visually demonstrates the kinematics of the system, including the motion of the bodies, the velocity and acceleration vectors, and the angular velocities of Body 2 and Body 3 over time.

2.3 Code

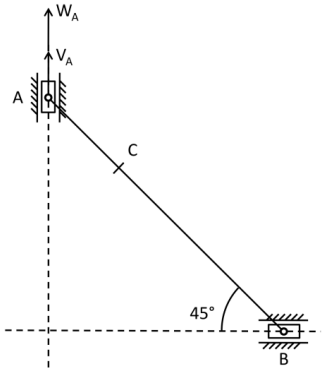
The full code solution is available on GitHub: GitHub Repository Link.

Task 3:

You should find:

1. simulate this mechanism (obtain all positions.) ($x_i(t)$, $y_i(t)$, where i is A, B, C point)
2. velocities for B, C (draw plots for magnitudes and show vectors on simulation);
3. accelerations for B and C (draw plots for magnitudes and show vectors on simulation);
4. draw a plot of angular velocity of body BA.

If $y_A(t) = 22.5 + 10sin(\frac{\pi}{5}t)$; $t = [0..10]$ $sec.$;
$AB = 45$, $BC = 30$.



Task 3
(Yablonskii (rus) K3)

In this simulation, we model the motion of a mechanical system consisting of three points: A, B, and C. Points A and B are connected by a rigid link

of length AB, and points B and C are connected by another link BC. The positions of points A, B, and C are calculated over time, and the resulting animation displays their motion.

Kinematic Model

The position of point A is given by the function:

$$y_A(t) = 22.5 + 10 \sin\left(\frac{\pi}{5} t\right)$$

where t is time. The link length AB is fixed at 45 units, and BC is fixed at 30 units.

For each time step, the positions of points A, B, and C are calculated as follows: - The position of point A is directly determined by $y_A(t)$, while the x-coordinate of point A is fixed at 0. - The position of point B is calculated by solving for x_B in the equation:

$$x_B^2 + y_A^2 = AB^2$$

where $AB = 45$. The value of x_B is determined using:

$$x_B = \sqrt{AB^2 - y_A^2}$$

- The position of point C is determined as a fraction along the line segment from A to B. Specifically, point C is placed $\frac{2}{3}$ of the way from A to B,

$$C = A + (B - A) \times \frac{2}{3}$$

Velocity and Acceleration Calculations

The velocities and accelerations of points B and C are calculated using central differences:

$$v_B = \frac{B(t + \Delta t) - B(t - \Delta t)}{2\Delta t}$$

$$v_C = \frac{C(t + \Delta t) - C(t - \Delta t)}{2\Delta t}$$

where Δt is the time step. Similarly, accelerations are computed as:

$$a_B = \frac{v_B(t + \Delta t) - v_B(t - \Delta t)}{2\Delta t}$$

$$a_C = \frac{v_C(t + \Delta t) - v_C(t - \Delta t)}{2\Delta t}$$

The angular velocity of the link AB is computed as:

$$\omega_{BA} = \frac{d\theta}{dt}$$

where θ is the angle between the vertical axis and the link AB, calculated using:

$$\theta = \arctan\left(\frac{x_B}{y_A - y_B}\right)$$

and ω_{BA} is the angular velocity of link AB.

Animation and Visualization

The animation is created using Matplotlib's FuncAnimation function. The positions of points A, B, and C are updated over time, and the links AB and BC are redrawn for each frame. The velocity and acceleration vectors for points B and C are displayed using quiver plots, with their magnitudes indicating the speed and acceleration at each point.

In addition to the animation, the program generates plots showing the velocity and acceleration magnitudes of points B and C, as well as the angular velocity of link AB over time.

2.4 Code

The full code solution is available on GitHub: GitHub Repository Link.