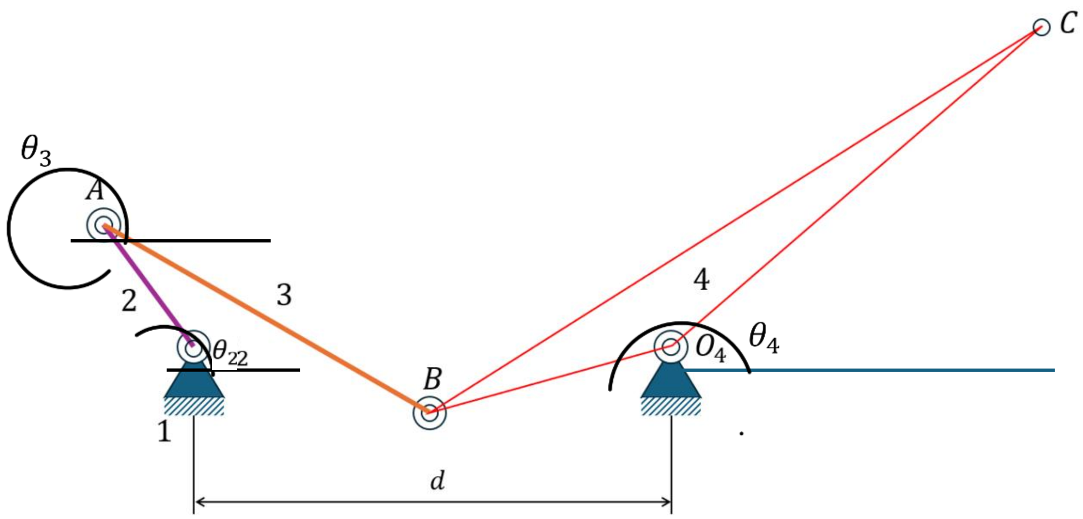
MS2105 Kinematics and Dynamics

Tugas Pemrograman Kinematika Komputasional

Nama : Raditya Alhamdika Fadhilah

NIM : 13123136

Dosen/Kelas : Prof. Ir. Andi Isra Mahyuddin, Ph.D./K01



A. Matriks-Matriks yang Dibutuhkan :

$$RL_2(\theta_2)=\begin{bmatrix} L_2\cos(\theta_2) \\ L_2\sin(\theta_2) \end{bmatrix}$$

$$RL_3(\theta_2,\theta_3) = \begin{bmatrix} L_2\cos(\theta_2) \\ L_2\sin(\theta_2) \end{bmatrix} + \begin{bmatrix} L_3\cos(\theta_3) \\ L_3\sin(\theta_3) \end{bmatrix}$$

$$RL_3(\theta_2,\theta_3) = RL_2(\theta_2) + \begin{bmatrix} L_3\cos(\theta_3) \\ L_3\sin(\theta_3) \end{bmatrix}$$

$$RO_4B(\theta_4) = \begin{bmatrix} d \\ 0 \end{bmatrix} + \begin{bmatrix} O_4B\cos(\theta_4) \\ O_4B\sin(\theta_4) \end{bmatrix}$$

Keterangan :

$RL_2(\theta_2)$ = Koordinat ujung L_2 sebagai fungsi θ_2
$RL_3(\theta_2,\theta_3)$ = Koordinat ujung L_3 sebagai fungsi θ_2 dan θ_3
$RO_4B(\theta_4)$ = Koordinat ujung O_4B sebagai fungsi θ_4

B. Persamaan Constraint

Koordinat L_3 dan O_4B akan selalu berakhir di titik yang sama, sehingga:

$$RL_3(\theta_2,\theta_3) = RO_4B(\theta_4)$$

$$\begin{bmatrix} L_2\cos(\theta_2) \\ L_2\sin(\theta_2) \end{bmatrix} + \begin{bmatrix} L_3\cos(\theta_3) \\ L_3\sin(\theta_3) \end{bmatrix} = \begin{bmatrix} d \\ 0 \end{bmatrix} + \begin{bmatrix} O_4B\cos(\theta_4) \\ O_4B\sin(\theta_4) \end{bmatrix}$$

$$L_2 \cos(\theta_2) + L_3 \cos(\theta_3) = d + O_4B \cos(\theta_4)$$

$$L_2 \sin(\theta_2) + L_3 \sin(\theta_3) = O_4B \sin(\theta_4)$$

$$L_2 \cos(\theta_2) + L_3 \cos(\theta_3) - d - O_4B \cos(\theta_4) = 0 \qquad (1)$$

$$L_2 \sin(\theta_2) + L_3 \sin(\theta_3) - O_4B \sin(\theta_4) = 0 \qquad (2)$$

Dengan menyelesaikan persamaan (1) dan (2) untuk setiap t dalam $\theta_2 = \omega t$, θ_3 dan θ_4 dapat ditentukan. Persamaan (1) dan (2) sangat rumit untuk diselesaikan secara aljabar, maka dari itu digunakan metode numerik untuk mendapatkan θ_3 dan θ_4 dari kedua persamaan tersebut.

Posisi batang O_4C dan batang BC dapat dihitung dengan mengetahui sudut antara O_4B dan O_4C mengingat batang O_4B- O_4C-BC bergerak dengan kecepatan dan percepatan sudut yang sama terhadap titik O_4.

Sudut O_4B- O_4C dapat dihitung dengan persamaan

$$\theta = \arccos\left(\frac{(O_4B)^2 - (O_4C)^2}{2 \cdot O_4B \cdot O_4B}\right)$$

Setelah data diagram kinematik posisi dan sudut setiap batang didapatkan, kecepatan dan percepatan anguler dapat dihitung dengan menggunakan metode approksimasi.

Kecepatan anguler dapat ditentukan dengan persamaan berikut:

$$\omega = \frac{\theta^n - \theta^{n-1}}{\Delta t}$$

Dengan :

ω = kecepatan anguler (rad/s)
θ^n= Sudut batang pada saat n (rad)
θ^{n-1} = sudut batang pada saat n-1 (rad)
Δt = rentang waktu antara θ^n dan θ^{n-1} (s)

Selanjutnya, percepatan anguler dapat dihitung dengan persamaan berikut

$$\alpha = \frac{\omega^n - \omega^{n-1}}{\Delta t}$$

Dengan :

α = percepatan anguler (rad/s)

ω^n= kecepatan anguler batang pada saat n (rad/s^2)

ω^{n-1} = Kecepatan anguler batang pada saat n-1 (rad/s^2)

Δt = rentang waktu antara ω^n dan ω^{n-1} (s)

C. Requirements according to Grashof's Equation

Grashof's criterion menyatakan bahwa suatu mekanisme empat batang memiliki setidaknya satu batang yang dapat berputar sepenuhnya apabila memenuhi ketentuan berikut:

$$S + L \leq P + Q$$

Dengan :

S = panjang batang terpendek

L = panjang batang terpanjang

P dan Q = panjang dua batang yang berada di antara S dan L

Pada mekanisme yang dianalisis, nilai-nilai panjang batang didefinisikan sebagai berikut:

$S = L_2$

$L = O_4B = 3.5L_2$

$P = L_3 = 2.9L_2$

$Q = d = 3L_2$

Dengan subtitusi panjang-panjang tersebut ke persamaan grashof, diperoleh :

$$L_2 + 3.5L_2 \leq 2{,}9L_2 + 3L_2$$

$$4.5L_2 \leq 5{,}9L_2$$

$$4.5 \leq 5{,}9$$

Hasil perhitungan ini menunjukkan bahwa mekanisme wiper yang dianalisis memenuhi kriteria Grashof. Dengan demikian, mekanisme ini dipastikan dapat berfungsi sesuai desainnya.

D. Program MATLAB

```
% Nama : Raditya Alhamdika Fadhilah
% NIM : 13123136
% Mata Kuliah : MS2105 KINEMATICS AND DYNAMICS
% Kelas : 01
% Dosen Pengajar : Prof. Ir. Andi Isra Mahyuddin, Ph.D.
% Deskripsi : Program analisis kinematika mekanisme winshield wiper
%             menggunakan matlab dengan crank(L2), coupler(L3) dan
%             wiper(L4). Dalam program ini menggunakan koordina referensi
%             utama (0,0) di ekor batang L2.

% Parameters
L2 = 50 + 136; % Length of Link 2
L3 = 2.9 * L2; % Length of Link 3
O4B = 1.2 * L2; % Distance O4 to B
O4C = 3.5 * L2; % Length of Link O4C
BC = 4*L2
d = 3 * L2; % Distance between pivots
Omega2 = 6; % Angular velocity of Link 2 (rad/s)
dt = 0.009; % Time step

% menghitung sudut O4B dan O4C
angleBC = acos((O4B^2 + O4C^2 - BC^2) / (2 * O4B * O4C));

% Initial conditions
t = 0; % Start time
Theta3_guess = (3/2)*pi; % Initial guess for Theta3
Theta4_guess = (3/2)*pi; % Initial guess for Theta4
Theta3_past = 0;
Theta4_past = 0;
Omega3_past = 0;
Omega4_past = 0;
Theta2_initial = pi/3;


% Initialize storage for plotting variables
time_array = [];
omega2_array = [];
omega3_array = [];
omega4_array = [];
angular_acceleration3_array = [];
angular_acceleration4_array = [];

% Initialize figure and subplots
figure;

% Subplot 1: Kinematics diagram
```

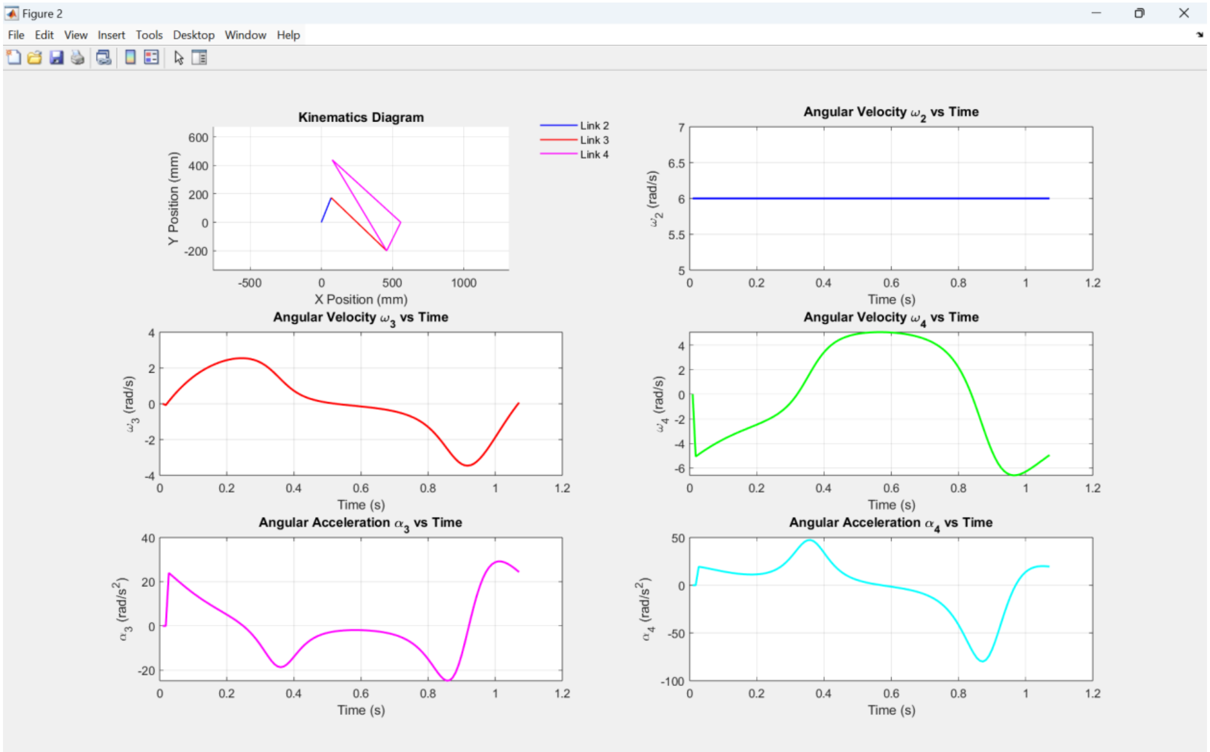
```matlab
subplot(3, 2, 1);
kinematics_axes = gca; % Get current axes
hold(kinematics_axes, 'on');
grid(kinematics_axes, 'on');
title('Kinematics Diagram');
xlabel('X Position (mm)');
ylabel('Y Position (mm)');
xlim([-0.01* L2, 0.8 * L2]);
ylim([-1.8 * L2, 3.6 * L2]);
axis equal;

% Initialize placeholders for kinematics links
link2 = plot(kinematics_axes, [0, 0], [0, 0], 'b-', 'LineWidth', 1); % Link 2
link3 = plot(kinematics_axes, [0, 0], [0, 0], 'r-', 'LineWidth', 1); % Link 3
linkO4B = plot(kinematics_axes, [0, 0], [0, 0], 'm-', 'LineWidth', 1); % Link
O4B
linkO4C = plot(kinematics_axes, [0, 0], [0, 0], 'm-', 'LineWidth', 1); % Link
O4C
linkBC = plot(kinematics_axes, [0, 0], [0, 0], 'm-', 'LineWidth', 1); % Link
O4C

% Add legend
legend([link2, link3, linkO4B], ...
       {'Link 2', 'Link 3', 'Link 4'}, ...
       'Location', 'northeast', 'Box', 'off','EdgeColor', 'none', 'Position',
[0.4, 0.85, 0.15, 0.1]);


% Subplots for angular variables
subplot(3, 2, 2); omega2_plot = plot(0, 0, 'b-', 'LineWidth', 1.5);
title('Angular Velocity \omega_2 vs Time'); grid on;
xlabel('Time (s)'); ylabel('\omega_2 (rad/s)');

subplot(3, 2, 3); omega3_plot = plot(0, 0, 'r-', 'LineWidth', 1.5);
title('Angular Velocity \omega_3 vs Time'); grid on;
xlabel('Time (s)'); ylabel('\omega_3 (rad/s)');

subplot(3, 2, 4); omega4_plot = plot(0, 0, 'g-', 'LineWidth', 1.5);
title('Angular Velocity \omega_4 vs Time'); grid on;
xlabel('Time (s)'); ylabel('\omega_4 (rad/s)');

subplot(3, 2, 5); alpha3_plot = plot(0, 0, 'm-', 'LineWidth', 1.5);
title('Angular Acceleration \alpha_3 vs Time'); grid on;
xlabel('Time (s)'); ylabel('\alpha_3 (rad/s^2)');

subplot(3, 2, 6); alpha4_plot = plot(0, 0, 'c-', 'LineWidth', 1.5);
title('Angular Acceleration \alpha_4 vs Time'); grid on;
xlabel('Time (s)'); ylabel('\alpha_4 (rad/s^2)');
```

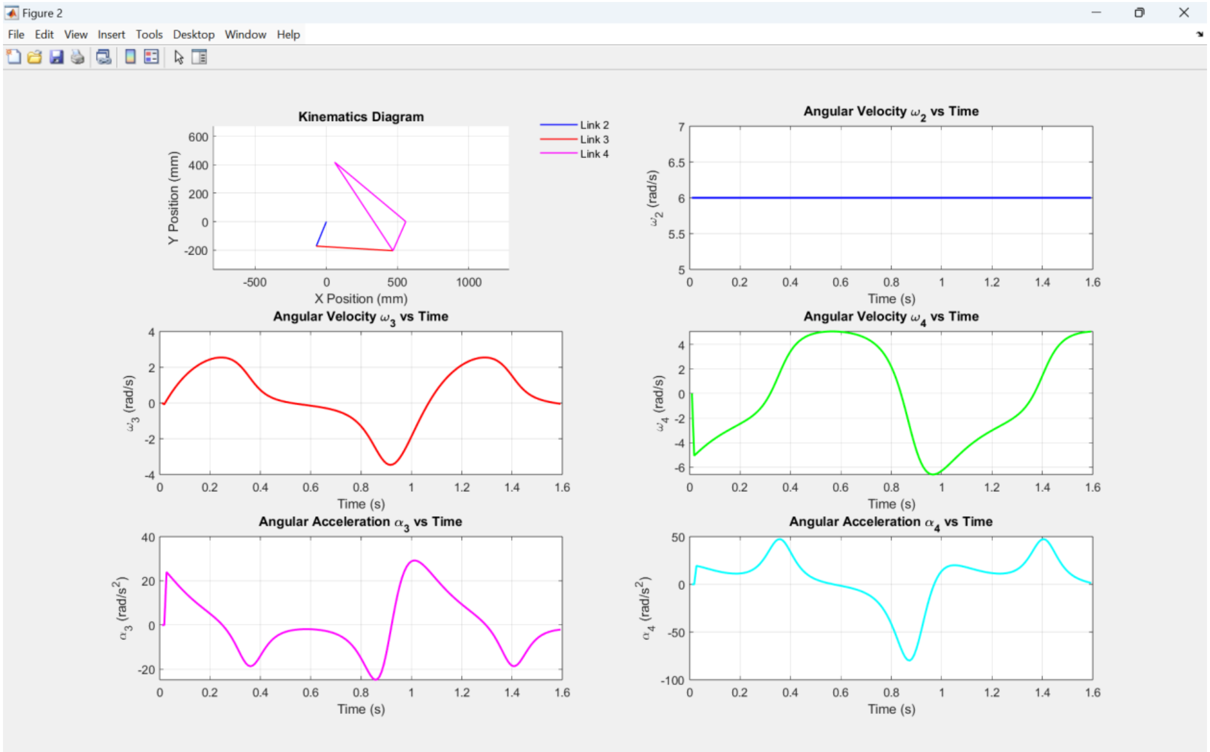
```
% Continuous simulation
i = 0; % Initialize iteration counter
max_time = 2; % Set maximum simulation time
while t < max_time
    % Update time and Theta2
    t = t + dt;
    Theta2 = Theta2_initial + Omega2 * t; % Angular position of Link 2

    % persamaan untuk mendapatkan theta3 dan theta4. Untuk setiap nilai t
    % dihitung secara numerik untuk mendapatkan nilai theta3 dan theta4.
    eqns = @(x) [
        L2 * cos(Theta2) + L3 * cos(x(1)) - d - O4B * cos(x(2)); % Equation 1
(X-axis)
        L2 * sin(Theta2) + L3 * sin(x(1)) - O4B * sin(x(2));     % Equation 2
(Y-axis)
    ];

    % Initial guess for [Theta3, Theta4]
    initial_guess = [Theta3_guess, Theta4_guess];  % Start with an initial
guess for Theta4 in the second quadrant

    % Use fsolve to solve the system numerically
    options = optimoptions('fsolve', 'Display', 'off');  % Turn off display
    solution = fsolve(eqns, initial_guess, options);

    % Extract Theta3 and Theta4 from the solution
    Theta3 = solution(1);
    Theta4 = solution(2);

    % menghitung kecepatan anguler dan percepatan anguler menggunakan
    % metode approksimasi
    if Theta3_past ~= 0 && Theta4_past ~= 0
        Omega3 = (Theta3 - Theta3_past) / dt;
        Omega4 = (Theta4 - Theta4_past) / dt;
    else
        Omega3 = 0; % Initial value for Omega3
        Omega4 = 0; % Initial value for Omega4
    end

    if Omega3_past ~= 0 && Omega4_past ~= 0
        angular_acceleration3 = (Omega3 - Omega3_past) / dt;
        angular_acceleration4 = (Omega4 - Omega4_past) / dt;
    else
        angular_acceleration3 = 0; % Initial value for Omega3
        angular_acceleration4 = 0; % Initial value for Omega4
    end

    % Persamaan semua batang sambungan
    R1 =[L2 * cos(Theta2);
```

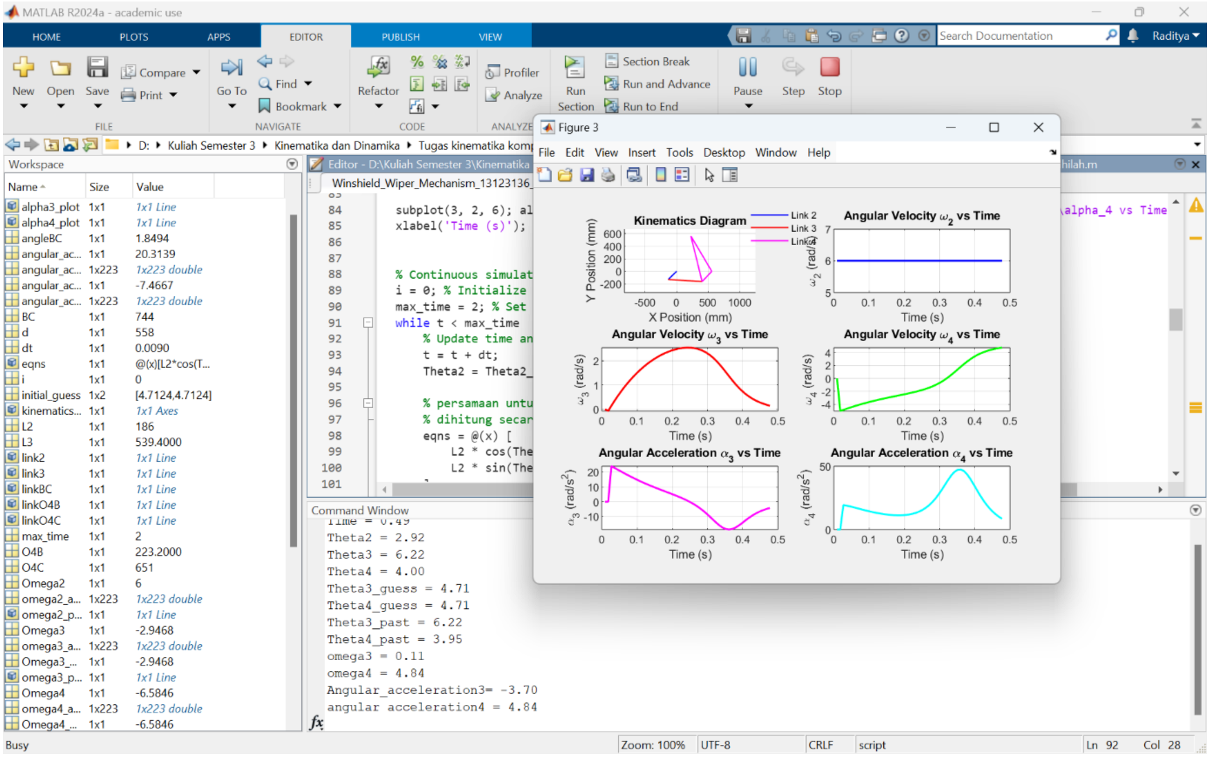
```
        L2 * sin(Theta2)];
    R3 = R1 + [L3 * cos(Theta3);
              L3 * sin(Theta3)];
    R4 = [d; 0] + [O4B * cos(Theta4);
                   O4B * sin(Theta4)];
    R5 = [d; 0] + [O4C * cos(Theta4 - angleBC);
                   O4C * sin(Theta4 - angleBC)];

    % plot sambungan-sambungannya
    set(link2, 'XData', [0, R1(1)], 'YData', [0, R1(2)]);
    set(link3, 'XData', [R1(1), R3(1)], 'YData', [R1(2), R3(2)]);
    set(linkO4B, 'XData', [d, R4(1)], 'YData', [0, R4(2)]);
    set(linkO4C, 'XData', [d, R5(1)], 'YData', [0, R5(2)]);
    set(linkBC, 'Xdata', [R4(1), R5(1)], 'Ydata', [R4(2), R5(2)])

    % input data-data ke dalam array untuk diplot
    time_array = [time_array, t];
    omega2_array = [omega2_array, Omega2];
    omega3_array = [omega3_array, Omega3];
    omega4_array = [omega4_array, Omega4];
    angular_acceleration3_array = [angular_acceleration3_array,
angular_acceleration3];
    angular_acceleration4_array = [angular_acceleration4_array,
angular_acceleration4];

     % Update data plots
    set(omega2_plot, 'XData', time_array, 'YData', omega2_array);
    set(omega3_plot, 'XData', time_array, 'YData', omega3_array);
    set(omega4_plot, 'XData', time_array, 'YData', omega4_array);
    set(alpha3_plot, 'XData', time_array, 'YData',
angular_acceleration3_array);
    set(alpha4_plot, 'XData', time_array, 'YData',
angular_acceleration4_array);



    % melihat nilai-nilai variabel di terminal
    fprintf('Time = %.2f\n', t);
    fprintf('Theta2 = %.2f\n', Omega2 * t);
    fprintf('Theta3 = %.2f\n', Theta3);
    fprintf('Theta4 = %.2f\n', Theta4);
    fprintf('Theta3_guess = %.2f\n', Theta3_guess);
    fprintf('Theta4_guess = %.2f\n', Theta4_guess);
    fprintf('Theta3_past = %.2f\n', Theta3_past);
    fprintf('Theta4_past = %.2f\n', Theta4_past);
    fprintf('omega3 = %.2f\n', Omega3);
    fprintf('omega4 = %.2f\n', Omega4);
    fprintf('Angular_acceleration3= %.2f\n', angular_acceleration3);
    fprintf('angular acceleration4 = %.2f\n', Omega4), angular_acceleration4;
```

```
    % Store the current values as the "previous" for next iteration
    Theta3_past = Theta3;
    Theta4_past = Theta4;
    Omega3_past = Omega3;
    Omega4_past = Omega4;



    % Pause for animation effect
    pause(dt);
    clc;
end
```

E. Plot Kinematik







File lengkap termasuk file MATLAB analisis kinematik ini bisa diakses di tautan berikut.

https://github.com/RadithyaAl/Kinematics-Analisys-of-Winshield-Wiper-with-Matlab