**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 black number on a white background

Description automatically generatedA black and white symbol

Description automatically generatedA diagram of a line with a point and a triangle

Description automatically generated with medium confidence



1. **Matriks-Matriks yang Dibutuhkan :**

=

Keterangan :

= Koordinat ujung L2 sebagai fungsi

= Koordinat ujung L3 sebagai fungsi dan

= Koordinat ujung sebagai fungsi

1. **Persamaan Constraint**

Koordinat L3 dan O4B akan selalu berakhir di titik yang sama, sehingga:

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

Posisi batang O4C dan batang BC dapat dihitung dengan mengetahui sudut antara O4B dan O4C mengingat batang O4B- O4C-BC bergerak dengan kecepatan dan percepatan sudut yang sama terhadap titik O4.

Sudut O4B- O4C dapat dihitung dengan persamaan

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:

Dengan :

= kecepatan anguler (rad/s)

= Sudut batang pada saat n (rad)

= sudut batang pada saat n-1 (rad)

= rentang waktu antara dan (s)

Selanjutnya, percepatan anguler dapat dihitung dengan persamaan berikut

Dengan :

= percepatan anguler (rad/s)

= kecepatan anguler batang pada saat n (rad/s2)

= Kecepatan anguler batang pada saat n-1 (rad/s2)

= rentang waktu antara dan (s)

1. **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:

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 = L2

L = O4B = 3.5L2

P = L3 = 2.9L2

Q = d = 3L2

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

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

1. **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

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

1. **Plot Kinematik**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

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

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