WIPER MECHANISM

PROJECT REPORT

Submitted by:
BATCH - 3
B.E.PRANAV KUMAAR : CB.EN.U4AIE20052
DIVI ESWAR CHOWDARY: CB.EN.U4AIE20012
DINESH KUMAR M.R : CB.EN.U4AIE20011
DABBRA HARSHA : CB.EN.U4AIE20010

of
2nd SEM B.Tech CSE-AI

For the Completion of
19PHY113 – COMPUTATIONAL ENGINEERING MECHANICS II
CSE - AI

AMRITA VISHWA VIDYAPEETHAM, ETTIMADAI

Submitted on:
18th July 2021

ACKNOWLEDGEMENT

Nammah Shivaya, firstly of all of us express our gratitude to Amma as finally we were able to finish our assignment that has been given by our Mechanics teacher to us.

Next, we would like to express our special thanks to our project guide and teacher Mr.Gopalakrishnan E.A who gave us the golden opportunity to do this wonderful project on the topic “Wiper Mechanism”, which in turn helped us in doing a lot of research in fields we were less familiar with, via which we learned so many new things. We are really thankful to them.

Lastly, we would also like to thank our parents and friends who helped us a lot in finalizing this project within the limited time frame and keeping us motivated throughout the process.

We would like to declare that we never knowingly discredited anyone of their work nor have we made efforts to plagiarize and for those whose work we referred to suitably equip ourselves to complete this project we acknowledge their effort and thank them for it.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

Symbol	Abbreviation/Explanation
WM	Wiper Mechanism
GUI	Graphical User Interface

1. INTRODUCTION

1.1 What is Wiper Mechanism?

A Mechanism is the part of a machine which contains two or more pieces arranged so that the motion of one compels the motion of the others.
A Wiper Mechanism (WM) is a specific type of mechanism with the defined purpose of removal of debris from a surface. Its most common application is to act as a windscreen wiper in vehicles to clean dirt or to clear the windscreen when it rains, other applications include solar panel wipers, window wipers, cleansing wipers etc. All the applications of WM follow the basic principle – A motor drives the mechanism by providing the input which is then transmitted appropriately which then drives the wiper arms which move along the surface effectively removing the debris.

1.2 What is the Aim of this project?

We are going to create a manipulatable simulation of a windscreen wiper and observe its functioning. We can then change various factors of the mechanism and observe their effects on the functioning. By doing so we can learn the optimal values of features that are compatible with real world applications. We also look to include other features like 1. error handling – to determine whether the WM can be made with the given input parameters, 2. file save – if we have experimented and found a configuration, we like to remember we can save the parameters of the same in a file of the user's choice and 3. We have included graphs of some parameters which the user can choose to view which would have been a real-world measurement of those parameters but in our case, we have used a perfect simulation.

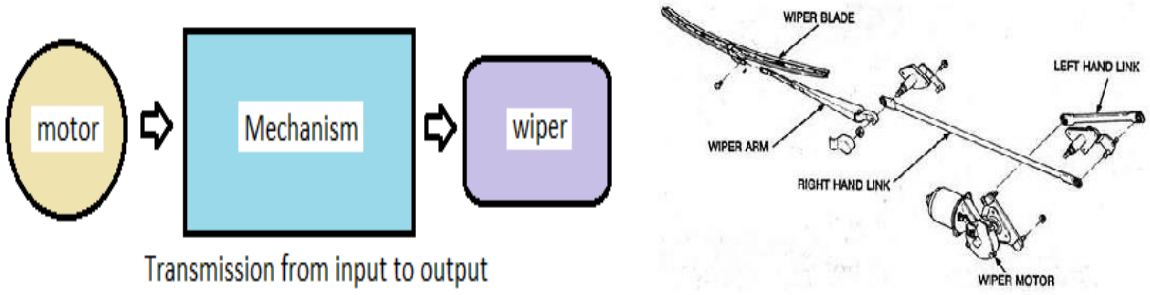
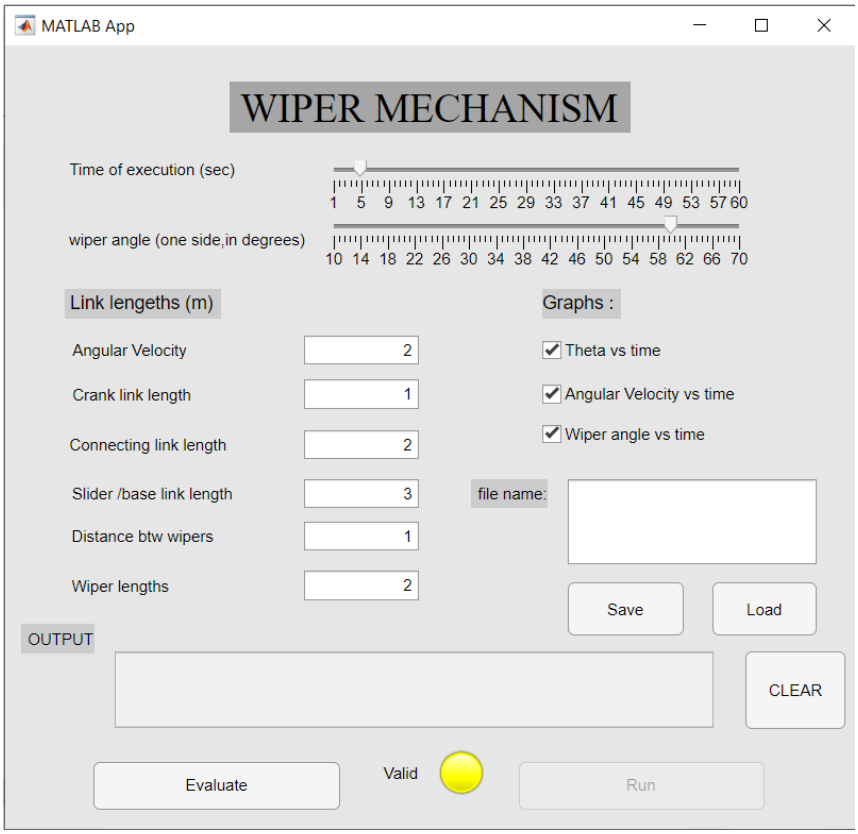


Figure 1.1 & 1.2 – Depicts the general idea & the parts of a mono-blade wiper

SOFTWARE REQUIRED:
MATLAB Appdesigner & image processing toolbox v11.2

2. IMPLEMENTATION

2.1 Simulation:



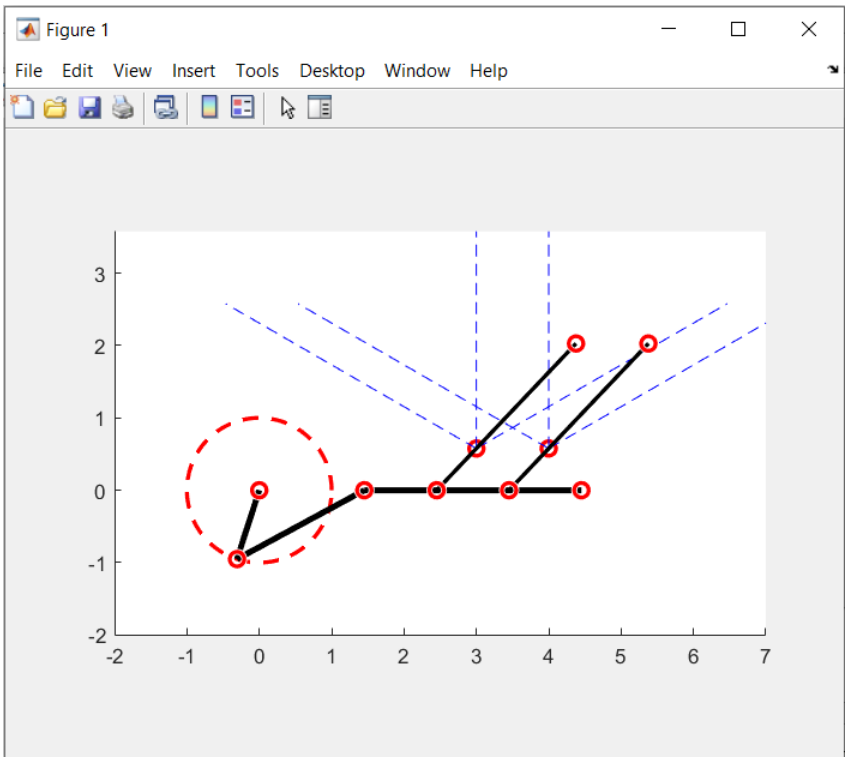


Figure 2.1 & 2.2 – Wiper mechanism application & simulation

2.2 Working Logic:

SIMULATION

We have used a slider crank mechanism to act as a WM by connecting the wiper arms to the slider and hinged them in-between. We have used basic linear equations to calculate the locations of the joints and ends of all links using the lengths as a parameter. Then we have used the points and drew and deleted appropriate lines with respect to each iteration.

GUI

For the purpose of building a GUI we have used an inbuilt software feature of MATLAB called Appdesigner. We have used the following GUI components in our project - 1. Sliders, 2. Check Boxes, 3. Text Fields, 4. Numeric Fields, 5. Lamp and 6. Plot Figures. As for their functionalities we have added call back functions.

ERROR HANDLING

We have restricted the type and range of input within the permissible limit for the parameters that we decided fit. For other types of errors, we have hard coded the App to evaluate the possible errors and display them as a warning/error in the 'OUTPUT Text Field'. This part of the process is performed as the function of the 'Evaluate Button'. When the 'Evaluate Button' is run the following outcomes are possible:

1. Error found – 'Lamp' turns Red and the state of the 'Run Button' remains disabled, the error message is displayed on the 'OUTPUT Text Field'.
2. Minor Errors – 'Lamp' turns Green and the state of the 'Run Button' changes to enabled, the simulation runs and a corresponding warning message is displayed on the 'OUTPUT Text Field'.
3. No Errors – 'Lamp' turns Green and the state of the 'Run Button' changes to enabled, the simulation runs.

FILE SAVE

We have added a Button called Save and Load which are responsible for this functionality.
Saving - We can save a file that contains a table of all the parameter values in any format that we prefer by inputting the "Name.Extension" in the 'file name Field'.
Loading – We can only load a file of type excel spreadsheets.

The ‘CLEAR Button’ clears the ‘OUTPUT’ and ‘file name’ Fields.

GRAPHS

We can generate the following graphs:

1. Theta (Total Angle driven) vs Time
2. Angular Velocity vs Time
3. Wiper angle (wrt to vertical) vs Time

The generation of these graphs is optional to the user and is controlled by the state of the respective check boxes.

2.3 MATLAB Code:

*Only the corresponding callback code of each functional component of GUI is included.

EVALUATE BUTTON:

```
297         % Button pushed function: EvaluateButton
298         function EvaluateButtonPushed(app, event)
299             error = false;
300 
301             base_val = app.SliderbaselinklengthEditField.Value;
302             width = app.DistancebtwwipersEditField.Value;
303             a = app.CranklinklengthEditField.Value;
304             b = app.ConnectinglinklengthEditField.Value;
305             wip_range = app.wiperangleonesideindegreesSlider.Value;
306             w = app.AngularVelocityEditField.Value;
307 
308             if(width > base_val)
309                 app.OUTPUTTextArea.Value = ['ERROR : Cannot design physical system with given parameters
310                     'Recommed to use smaller value for width'];
311                 error = true;
312             end
313 
314             if(a>=b)
315                app.OUTPUTTextArea.Value = ['ERROR : Cannot design physical system with given parameters
316                     'Recommed to use larger value of connecting link/ smaller value of crank link'];
317                 error = true;
318             end
319 
320             if(wip_range<50)
321                app.OUTPUTTextArea.Value = ['WARNING: non-ideal values of hinge height will be obtained
322                     'Recommed to use larger value wiper angle'];
323                pause(2)
324             end
325 
326             if(w == 0)
327                 app.OUTPUTTextArea.Value = ['WARNING: cannot observe any motion because w = 0
328                     'Recommed to use w != 0'];
329                pause(2)
330             end
331             %
332             %
333             if(error == false)
334                app.ValidLamp.Color = 'g';
335                app.OUTPUTTextArea.Value = 'Proceed to run simulation';
336                app.RunButton.Enable = true;
337                pause(3);
338                app.ValidLamp.Color = 'y';
339                app.RunButton.Enable = false;
340             elseif(error == true)
341                app.ValidLamp.Color = 'r';
342                pause(3);
343                app.ValidLamp.Color = 'y';
344             end
345         end
```

SAVE BUTTON:

```
        % Button pushed function: SaveButton
        function SaveButtonPushed(app, event)
            a = app.CranklinklengthEditField.Value;
            b = app.ConnectinglinklengthEditField.Value;
            base_val = app.SliderbaselinklengthEditField.Value;
            width = app.DistancebtwwipersEditField.Value;
            l = app.WiperlengthsEditField.Value;
            wip_range = app.wiperangleonesideindegreesSlider.Value;
            T = app.TimeofexecutionsecSlider.Value;
            w = app.AngularVelocityEditField.Value;
            graph1 = app.ThetavstimeCheckBox.Value;
            graph2 = app.AngularVelocityvstimeCheckBox.Value;
            graph3 = app.WiperanglevstimeCheckBox.Value;

            % Create a table with the data and variable names
            Table = table(a, b, base_val, width, l, wip_range, T, w, graph1, graph2, graph3, ...
                'VariableNames', { 'crank length', 'connencting link length', 'slider link length', ...
                'wiper width', 'wiper length', 'wiper angle range', 'time of execution', ...
                'angular velocity', 'graph1', 'graph2', 'graph3'});
            value = app.filenameEditField.Value;
            filename = value;

            % Write data to text file
            writetable(Table, filename);
        end
```

LOAD BUTTON:

```
        % Button pushed function: LoadButton
        function LoadButtonPushed(app, event)
            value = app.filenameEditField.Value;
            filename = value;

            [num,txt,raw] = xlsread(filename);
            app.CranklinklengthEditField.Value = num(1);
            app.ConnectinglinklengthEditField.Value = num(2);
            app.SliderbaselinklengthEditField.Value = num(3);
            app.DistancebtwwipersEditField.Value = num(4);
            app.WiperlengthsEditField.Value = num(5);
            app.wiperangleonesideindegreesSlider.Value = num(6);
            app.TimeofexecutionsecSlider.Value = num(7);
            app.AngularVelocityEditField.Value = num(8);
            app.ThetavstimeCheckBox.Value = num(9);
            app.AngularVelocityvstimeCheckBox.Value = num(10);
            app.WiperanglevstimeCheckBox.Value = num(11);

        end
    end
```

CLEAR BUTTON:

```
        % Button pushed function: CLEARButton
        function CLEARButtonPushed(app, event)
            app.OUTPUTTextArea.Value = '';
            app.filenameEditField.Value = '';
        end
```

RUN BUTTON:

```
        % Button pushed function: RunButton
        function RunButtonPushed(app, event)

            % length in m
            a = app.CranklinklengthEditField.Value;      % crank link length
            b = app.ConnectinglinklengthEditField.Value; % connencting link length

            % wiper base parameters
            base_val = app.SliderbaselinklengthEditField.Value;
            width = app.DistancebtwwipersEditField.Value;
            buffer = (base_val-width)/2;

            % wiper parameters
            l = app.WiperlengthsEditField.Value;
            wip_range = app.wiperangleonesideindegreesSlider.Value;
            h = a/tand(wip_range);
            k = 2*l;

            % time of execution
            T = app.TimeofexecutionsecSlider.Value;
            iter = T*10;

            % the angular speed of the crank in rad/s
            w = app.AngularVelocityEditField.Value;


            % parameters of the plot
            neg_x = -(a + 1);
            pos_x = (a + b + base_val + 1);
            neg_y = -(a + 1);
            if((l + h) > a)
                pos_y = (l + h + 1);
            else
                pos_y = a+1;
            end
            axis(gca, 'equal'); % the aspect ratio
            axis([neg_x pos_x neg_y pos_y]);  % the limits


            % fixed joint
            P1 = [0 0];

            % the point P2s trajectory
            P2_traj = viscircles([0 0],a,'Linestyle','--');

            % locations of hinges
            O = [b 0];
            H1 = O + [buffer h];
            H2 = O + [(buffer + width) h];

            H1_circ = viscircles(H1,0.1);
            H2_circ = viscircles(H2,0.1);

            % locations of expected extermities
            M1 = H1 + k*[0 1];
            M2 = H2 + k*[0 1];
            ER1 = H1 + k*[sind(wip_range) cosd(wip_range)];
            EL1 = H1 + k*[-sind(wip_range) cosd(wip_range)];
            ER2 = H2 + k*[sind(wip_range) cosd(wip_range)];
            EL2 = H2 + k*[-sind(wip_range) cosd(wip_range)];

```

```
% drawing those lines
PM1 = line([H1(1) M1(1)],[H1(2) M1(2)],'Color','blue','Linestyle','--');
PM2 = line([H2(1) M2(1)],[H2(2) M2(2)],'Color','blue','Linestyle','--');
PER1 = line([H1(1) ER1(1)],[H1(2) ER1(2)],'Color','blue','Linestyle','--');
PEL1 = line([H1(1) EL1(1)],[H1(2) EL1(2)],'Color','blue','Linestyle','--');
PER2 = line([H2(1) ER2(1)],[H2(2) ER2(2)],'Color','blue','Linestyle','--');
PEL2 = line([H2(1) EL2(1)],[H2(2) EL2(2)],'Color','blue','Linestyle','--');

% for other plots
w_value = w*ones(iter);
time = zeros(iter);
wiper_ang = zeros(iter);
theta_value = zeros(iter);

for t = 1:iter % t is in 10^-1s
    theta = (t/10)*w;

    theta_value(t) = theta;
    time(t) = t/10;

    % the point P2
    P2 = a*[cos(theta) sin(theta)];

    % angle beta
    beta = asin((a*sin(theta))/b);
    % the point P3
    P3 = [((a*cos(theta)) + (b*cos(beta))) 0];
    P4 = [((a*cos(theta)) + (b*cos(beta)) + base_val) 0];
    B1 = [((a*cos(theta)) + (b*cos(beta)) + buffer) 0];
    B2 = [((a*cos(theta)) + (b*cos(beta)) + buffer + width) 0];

    % wiper position
    I1 = H1 - B1;
    W1_ang = atan(I1(1)/I1(2));
    W1 = H1 + l*[sin(W1_ang) cos(W1_ang)];
    wiper_ang(t) = W1_ang;

    I2 = H2 - B2;
    W2_ang = atan(I2(1)/I2(2));
    W2 = H2 + l*[sin(W2_ang) cos(W2_ang)];


    % crank line
    crank = line([P1(1) P2(1)],[P1(2) P2(2)],'Color','black','LineWidth',3);
    % slider line
    slider = line([P2(1) P3(1)],[P2(2) P3(2)],'Color','black','LineWidth',3);
    % base line
    base = line([P3(1) P4(1)],[P3(2) P4(2)],'Color','black','LineWidth',3);


    % wipers
    wiper1 = line([B1(1) W1(1)],[B1(2) W1(2)],'Color','black','LineWidth',2);
    wiper2 = line([B2(1) W2(1)],[B2(2) W2(2)],'Color','black','LineWidth',2);

    % highlighting P1 and P2
    P1_circ = viscircles(P1,0.1);
    P2_circ = viscircles(P2,0.1);
    P3_circ = viscircles(P3,0.1);
    P4_circ = viscircles(P4,0.1);
    B1_circ = viscircles(B1,0.1);
    B2_circ = viscircles(B2,0.1);
```

```
                W1_circ = viscircles(W1,0.1);
                W2_circ = viscircles(W2,0.1);

                % the interval to update the plot
                pause(0.001);

                % delete the previous instantances
                delete(crank);
                delete(slider);
                delete(base);
                delete(wiper1);
                delete(wiper2);
                delete(P1_circ);
                delete(P2_circ);
                delete(P3_circ);
                delete(P4_circ);
                delete(B1_circ);
                delete(B2_circ);
                delete(W1_circ);
                delete(W2_circ);
            end

            delete(P2_traj);
            delete(H1_circ);
            delete(H2_circ);
            delete(PER1);
            delete(PEL1);
            delete(PER2);
            delete(PEL2);
            delete(PM1);
            delete(PM2);


            close all;
            pause(0.5);


            graph1 = app.ThetavstimeCheckBox.Value;
            if(graph1 == 1)
                theta_value = rad2deg(theta_value);

                figure
                axis(gca, 'equal');
                plot(time, theta_value, 'Color', 'red', "LineWidth", 3);
                title('Theta vs Time');
                xlabel('Time in seconds');
                ylabel('Theta in degrees');
                pause(2);
            end

            graph2 = app.AngularVelocityvstimeCheckBox.Value;
            if(graph2 == 1)
                figure
                axis(gca, 'equal');
                plot(time, w_value, 'Color', 'black', "LineWidth", 3);
                title('Angular Velocity vs Time');
                xlabel('Time in seconds');
                ylabel('Angular Velocity in radians/second');
                pause(2)
            end

            graph3 = app.WiperanglevstimeCheckBox.Value;
            if(graph3 == 1)
                wiper_ang = rad2deg(wiper_ang);


                figure
                axis(gca, 'equal');
                plot(time, wiper_ang, 'Color', 'blue', "LineWidth", 3);
                title('Wiper angle vs Time');
                xlabel('Time in seconds');
                ylabel('Angular Velocity in degrees');
                pause(2);
            end



            app.OUTPUTTextArea.Value = 'Simulation ran successfully';
            pause(3);
            app.OUTPUTTextArea.Value = '';
        end

        % Value changed function: AngularVelocityEditField
        function AngularVelocityEditFieldValueChanged(app, event)
            % value = app.AngularVelocityEditField.Value;
        end
```

GRAPHS:

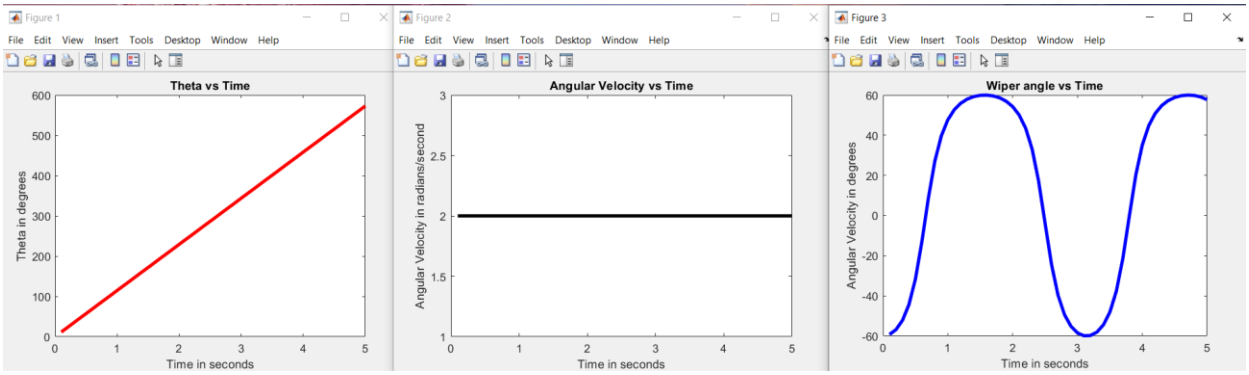


Figure 2.3 – All graphs for an example simulation

3. THEORETICAL AND PHYSICAL ASPECTS

3.1 Background Math:

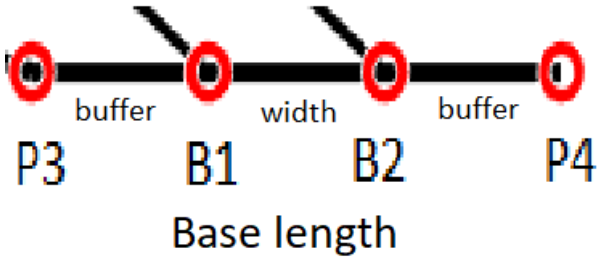


Figure 3.1 – Labelled diagram of wiper base link/ slider

End Buffer length = (Base length – Distance between wiper arms)/2

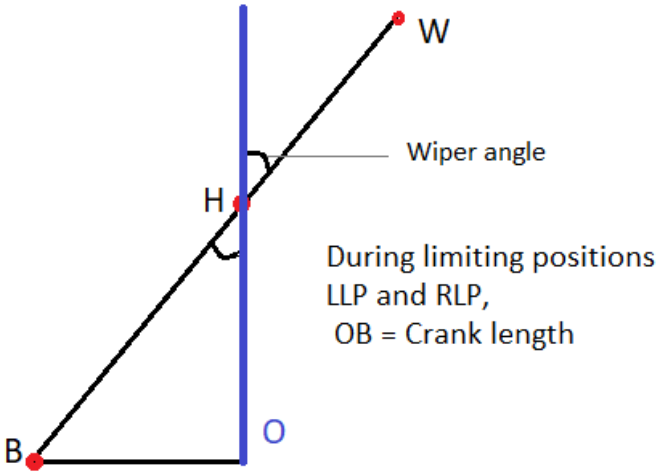


Figure 3.2 – Depicts a part of the mechanism

Hinge height = Crank link length / tan(maximum wiper angle)

Number of iterations = time * 10

Reason: We can generate the system 1/10 of a second makes the simulation smooth.

Consider,

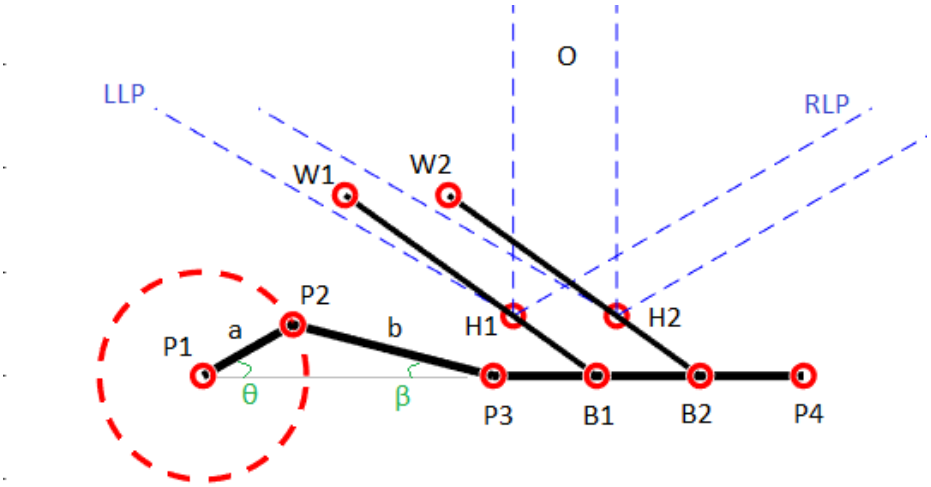


Figure 3.3 – Labelled diagram of WM

In this image LLP and RLP depict Left Limiting Position and Right Limiting Position of the wiper arms

P1 = 0 î + 0 ĵ - origin

Then,
P2 = a*cos(θ) î + a*sin(θ) ĵ

β = asin((a*sin(θ))/b) - sine law

Then,
P3 = (a*cos(θ)) + (b*cos(β)) î + 0 ĵ
P4 = ((a*cos(θ)) + (b*cos(β)) + base value) î + 0 ĵ
B1 = ((a*cos(θ)) + (b*cos(β)) + buffer) î + 0 ĵ
B2 = ((a*cos(θ)) + (b*cos(β)) + buffer + width) î + 0 ĵ

O = b î + 0 ĵ - neutral position of WM
H1 = O + buffer î + h ĵ
H2 = O + (buffer + width) î + h ĵ

Then,
W1 = H1 + l*sin(W1_ang) î + l*cos(W1_ang) ĵ
W2 = H2 + l*sin(W2_ang) î + l*cos(W2_ang) ĵ

3.2 Different types of WMs

The classification of WM can be done on the basis of two factors, the number of wiper arms and the motion that it performs.

Classification based on number of wiper arms:

1. Mono-blade/Single-arm – this class contains WMs that act on only one wiper arm, these are primarily found on rear windows where the utility is minimum and maximum area is obtained.
2. Dual-blade/Double-arm – this class contains WMs that act on two wiper arms.
3. Mutli-blade/multi-arm – this class contains WMs that act on more than two wiper arms. Applications of this class are very rare as they are only used in specific customized designs and are inefficient.

Classification based on motion:

The motion of single blade is based on the angle it is built to cover ignoring special mechanical motions.

Type Name	Motion description	Rank on effectivity as windscreen wipers
Pivot/Radial Synchronous	The motion of both the arm are in the same direction, these are generally by achieved the same driving link for both arms.	2
Sequential sweep	The motion of both the arm are in the opposite directions, these are generally by achieved the different driving link for both arms	3
Pantograph system	The wiper blades are pointing in constant direction but the arms are used to provide perpendicular motion	1

Table 3.1 – Types of Wiper Mechanisms

The design boundary between the different types is vague as they can be modified by small mechanical changes and each of these types can be applied in different positions with respect to the plane of application i.e., left/right/top/bottom. The different types of mechanism can

also be modified by the consideration input source and also the mechanism used to transfer the source to the arms e.g., inline slider, offset slider, uni-link etc.
The terms arms and blade represent two different parts of the WM, in some WM the blade and arm are combined.

3.3 Design Analysis:

MATERIAL:

The first part of building a mechanism is to consider the material used in the synthesis of parts of it.
Wiper- the wiper blades are made from natural rubber of synthetic compounds as it is required to not cause damage to the surface. Some rubber blades are composites of soft rubber on the wiping edge and firm rubber that supports the wiping edge in the rest of the blade. There are three types of wiper blades: conventional, flat and hybrid.
Mechanism- most components of the mechanism are made out of aluminum or metallic alloys with similar properties as it is durable, light weight and cheap. When synthesizing the total mechanism other factors like stress, strain, friction etc. are taken into account.
Surface- Almost all of the surfaces are 2D excluding some 3D exceptions. All surfaces are smooth (with no irregularities). The material of the surface is composed of reinforced/laminated safety glass, this type of glass consists of polycarbonate sheet sandwiched between two layers of glass.
Debris- The material composite of debris is variable. The physical state of the debris does not matter in case of the wiper mechanism.

METHOD:

The second part of building the mechanism is to consider the method used in design.
We can see the different types of wipers as discussed in section 3.2, in consideration of factors like cost, efficiency and purpose we can select the appropriate method of application corresponding to the problem.
In our case we found it best to simulate a bottom-middle pivoted synchronous dual arm wiper as an angle adjusted version of this is the most widely used application of WM.
To implement this used the inline slider-crank mechanism, we did this as the visualization will be better in comparison to an offset slider-crank mechanism, the difference being modification of math equation to suit the offset crank mechanism.

The amount of energy required to drive each mechanism greatly depends on the transmission loss of the mechanism. Factors like number of links used, their orientation and friction determine transmission loss.

3.4 Error Analysis:

The possible errors that can arise are:

1. Irregular input supply/motor malfunctioning
2. Inefficient transmission/design choice
3. Wiper friction
4. Irregular surface
5. External/human/random errors

The solution to 1-4th errors is to systematically debug the prototype and remake/change part that is causing the error. The 5th error cannot be control but will rectify itself when all other steps are verified.

4. CONCLUSION & FUTRUE SCOPE

We have successfully been able to complete a simulation of a working prototype of synchronous dual arm wiper that is able to correctly recreate the features we aimed to achieve to a satisfactory degree.

The existing windshield wiper washing system creates a temporary blindness when it is operated. two different operations (wiper and washing) may affect the view of driver while travelling. So, to avoid this horizontal wiper will be used to clean the entire wind shield and washer system will be through it. Another approach is to add a rain sensor, so the driver doesn't need to worry about the intensity and can focus on the driving due to automatic speed control wipers and when the rain keeps varying for skyscrapers then the wind shield can be cleaned without using any additional water.

BIBLIOGRAPHY

https://www.google.com – for definitions and information

ARTICLES

https://www.a1-windscreens.co.uk/news/windscreen-wiper-movements/ - types of WM

https://blog.autokartz.com/how-to-check-car-wiper/ - car wiper details

FIGURES

Figure 1.2 - https://blog.autokartz.com/how-to-check-car-wiper/

Made using Paint App