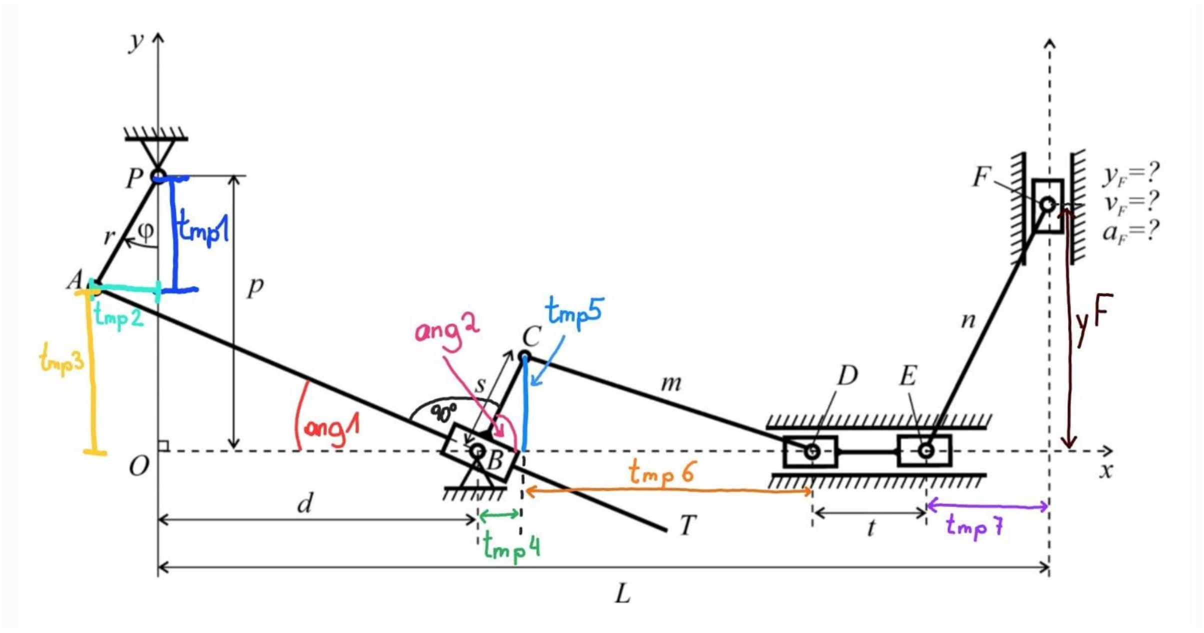
Section 1



The first step to calculate the vertical position of slider F is to address the dependent variable involved in the mechanism, which is the position of the crank AB with respect to a change in angle phi, φ. Simple right-angled triangle geometry (Stroud & Booth, 2007) can be used to compute the positional values of the crank end A relative to P:

yA = r*cos(phi) = tmp1
xA= r*sin(phi) = tmp2

Now the angle OBA, denoted ang1, must be calculated, which is done using right-angled triangles once again. The coordinate values for A need to be relative to another vertex of the triangle forming the trigonometric relationship (Jordan & Smith, 2010), hence tmp1 is translated to tmp3 = p-tmp1 to represent the vertical distance between A and O. Also, tmp2 is added to d to represent the horizontal distance between B and A. With these translations, ang1 is calculated using the 'TOA' ratio:

ang1 = arctan(tmp3/(d+tmp2))

Now, angle CBD = ang2 can be calculated due to the precedent that the CD linkage is always perpendicular, and so ABC = π/2 rad. OD is a straight line, hence ang1+ABC+ang2 = π rad. This can be rewritten as an equation to solve ang2:

ang2 = (pi/2)-ang1

From here, the coordinates of C relative to B can be calculated using right-angled trigonometry.

xC = s*cos(ang2) = tmp4
yC = s*sin(ang2) = tmp5

The vertical position of C is then used to determine the horizontal position of D using Pythagoras theorem:

xD = sqrt(m^2-tmp5^2) = tmp6

Now, all the distances along the x axis between O and F are all known, since the DE linkage is provided. Therefore, the remaining x component along L, between E and F, can be used in conjunction with pythagoras theorem to find yF:
tmp7 = L-t-tmp6-tmp4-d

yF = sqrt(n^2-tmp7^2)

Computing that in matlab works as followed:

```
% Define the range for x with steps of 0.001 radians
phi = (0:0.001:2*pi);

% define temporary variables and the function y
tmp1 = r * cos(phi);
tmp2 = r * sin(phi);
tmp3 = p - tmp1;
ang1 =  atan(tmp3./(d + tmp2));
ang2 = (pi/2) - ang1;
tmp4 = s * cos(ang2);
tmp5 = s * sin(ang2);
tmp6 = sqrt(m^2 - tmp5.^2);
tmp7 = L - t - tmp6 - tmp4 - d;

% Calculate the function y
yF = sqrt(n^2 - tmp7.^2);
```

Section 2

To find the maximum and minimum values for yF, an array must be set up for values of phi named ‘Maximum’ and ‘Minimum’.

By starting with the first value in the array, the code establishes a baseline for comparison. It then iterates through each element of the phi array with steps of 0.001 radians. This can ensure that each element of yF is checked, maintaining consistency with the step size of the phi array. The code checks if the current element of yF is greater than the stored maximum value, or smaller than the stored minimum. If the current yF value is greater, it updates ‘Maximum’ to that value and stores the corresponding angle (phi) in ‘Maximum_x’.

The converse is performed to also reach the minimum value. This ensures precise tracking of the extreme values and their corresponding angles, and is iterated through a ‘for loop’ (Etter, 2015):

```
% Iterate through yF values and find maximum and minimum values
for i = 1:length(phi)
    if yF(i) > Maximum
       Maximum = yF(i); % Update max value if current yF value is larger
       Maximum_x = phi(i); % Store corresponding x value
    end
   if yF(i) < Minimum
        Minimum = yF(i); % Update min value if current yF value is smaller
        Minimum_x = phi(i); % Store corresponding x value
   end
end
```

Section 3

To calculate the velocity of slider F, the central difference method is used. First, an array vF is initialized that has the same size as the array phi. Then a loop is created that iterates through the elements of phi, starting at 2 and ending at the second-to-last element: (Méndez-Vilas, 2005)

```matlab
% Initialize array to store velocity values
vF = zeros(size(phi));

% Calculate velocity using central difference method
for i = 2:length(phi)-1
    vF(i) = (yF(i+1) - yF(i-1)) / (phi(i+1) - phi(i-1));
end
```

That is vital, as both the previous and the next point have to be part of the array for this method to work. The central difference method takes the difference between the yF values at surrounding positions and divides them by the corresponding position difference from phi to determine the velocity.

As point 1 and the endpoint were left out of the loop, we need to determine them separately, by using a forward difference for point 1 and a backward difference for the last point. The last step to get the actual velocity is to multiply it by the angular velocity, which is 2m/s and is given the variable ‘w’ as previous in the code:

```matlab
% Handle boundary points (endpoints)
vF(1) = (yF(2) - yF(1)) / (phi(2) - phi(1));
vF(end) = (yF(end) - yF(end-1)) / (phi(end) - phi(end-1));
% Multiply by angular velocity to get the actual velocity
vF = w * vF;
```

Section 4

Same as for the velocity, the acceleration points are stored in an array called aF, with a size of phi. Similar to the solution for obtaining the velocity, the code divides the difference in velocities at surrounding positions by the corresponding difference in positions from phi to calculate the acceleration at each position.
As the iteration starts at 3 and ends at end-2, the missing values have to be calculated or left out. (Méndez-Vilas, 2005)

Point 2 and end-1 are calculated using the forward and backward difference, however point 1 and the endpoint are excluded from the function, as there is insufficient data to calculate them.
To obtain the actual accelerations, it needs to be multiplied by w.

```matlab
% Initialize array to store acceleration values
aF = zeros(size(phi));
% Calculate acceleration using central difference method, excluding endpoints
for i = 3:length(phi)-2
    aF(i) = (vF(i+1) - vF(i-1)) / (phi(i+1) - phi(i-1));
end

%calculate the two endpoints for this function
aF(2) = (vF(3) - vF(2)) / (phi(3) - phi(2));
aF(end-1) = (vF(end-1) - vF(end-2)) / (phi(end-1) - phi(end-2));

%exclude startpoint and last endpoint as there isnt enough information to
%calculate them
aF([1, end]) = NaN;
aF = aF * w;
```

Section 5:
To obtain the velocity 0 points, a sign change between subsequent points has to be detected.
That is done by using a for loop and an if loop that checks if the current value vF(i) is negative and the next element vF(i+1) is positive, or vice versa. If this condition is true, it means there’s a sign change and a zero point in the velocity function.
Those values are then converted to their actual phi values, as those are the values needed.
We are required to use an absolute error of 0.0005, however as that aligns with the absolute error matlab uses, no specific modifications are necessary for the code.

```
%-------------------velocity 0 points-------------------
% Initialize array to store indices of zero points
velocity_0 = [];

% Detect sign changes in vF array
for i = 1:length(phi)-1
    if (vF(i) < 0 && vF(i+1) > 0) || (vF(i) > 0 && vF(i+1) < 0)
        velocity_0(end+1) = i; % Store index of zero point
    end
end

% Convert zeros to corresponding phi values
velocity_0 = phi(velocity_0);
```

References

Stroud, K., & Booth, D. (2007). *Engineering mathematics*. Palgrave Macmillan.

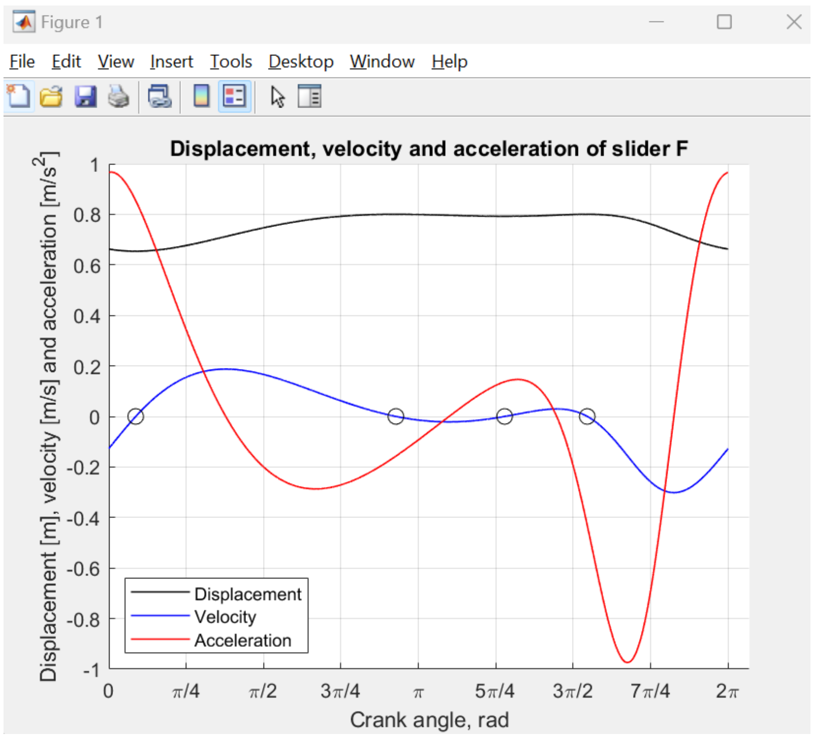
Etter, D. M. (2011). *Introduction to Matlab*. Pearson.

Jordan, D. W., & Smith, P. (2010). *Mathematical techniques: An introduction for the engineering, physical, and mathematical Sciences*. Oxford University Press.

Méndez-Vilas, A. (2005). *Recent advances in multidisciplinary applied physics, science direct.* Available at: https://www.sciencedirect.com/topics/mathematics/central-difference (Accessed on: 3.3.2024)

Results

This is what the results window looks like. We have included a legend for easier interpretation of the graph.



Slider F has the following minimum and maximum heights, along with corresponding crank angles. The velocity 0 points are shown as well.

```
The maximum height of Slider F (in m):
    0.8000

The corresponding crank angle (in rad):
    2.9140

The minimum height of slider F (in m):
    0.6539

The corresponding crank angle (in rad):
    0.2730

The points of zero velocity (in rad):
    0.2730    2.9130    4.0160    4.8540
```

Matlab Code (Full)

```
clear; clc;
%===================== INPUT DATA ==============================

r=0.30; % length of the crank PA in m
m=0.70; % length of the ink CD in m
p=0.55; % length of the distance OP in m
d=0.85; % length of the distance OB in m
s=0.65; % length of the distance CB in m
w=2; % angular velocity of the crank in rad/s
n=0.80; % length of the link EF, m
t=0.20; % length of the distance DE, m
L=2.0; % length of the distance L in m

%====================== COMPUTATION
==================================

% Define the range for x with steps of 0.001 radians
phi = (0:0.001:2*pi);

% define temporary variables and the function y
tmp1 = r * cos(phi);
tmp2 = r * sin(phi);
tmp3 = p - tmp1;
ang1 =  atan(tmp3./(d + tmp2));
ang2 = (pi/2) - ang1;
tmp4 = s * cos(ang2);
tmp5 = s * sin(ang2);
tmp6 = sqrt(m^2 - tmp5.^2);
tmp7 = L - t - tmp6 - tmp4 - d;

% Calculate the function y
yF = sqrt(n^2 - tmp7.^2);



%==================== Minimum and Maximum Values ====================

% Initialize variables to store maximum and minimum values
Maximum = yF(1); % Initialize variable
Minimum = yF(1);  % Initialize variable

% Iterate through yF values and find maximum and minimum values
for i = 1:length(phi)
    if yF(i) > Maximum
       Maximum = yF(i); % Update max value if current yF value is larger
       Maximum_x = phi(i); % Store corresponding x value
    end

   if yF(i) < Minimum
        Minimum = yF(i); % Update min value if current yF value is smaller
        Minimum_x = phi(i); % Store corresponding x value
   end
end
```

```
%===================== Velocity Calculation =====================

% Initialize array to store velocity values
vF = zeros(size(phi));

% Calculate velocity using central difference method
for i = 2:length(phi)-1
   vF(i) = (yF(i+1) - yF(i-1)) / (phi(i+1) - phi(i-1));
end

% Handle boundary points (endpoints)
vF(1) = (yF(2) - yF(1)) / (phi(2) - phi(1));
vF(end) = (yF(end) - yF(end-1)) / (phi(end) - phi(end-1));

% Multiply by angular velocity to get the actual velocity
vF = w * vF;


%===================== Acceleration Calculation =====================


% Initialize array to store acceleration values
aF = zeros(size(phi));
% Calculate acceleration using central difference method, excluding endpoints
for i = 3:length(phi)-2
   aF(i) = (vF(i+1) - vF(i-1)) / (phi(i+1) - phi(i-1));
end

%calculate the two endpoints for this function
aF(2) = (vF(3) - vF(2)) / (phi(3) - phi(2));
aF(end-1) = (vF(end-1) - vF(end-2)) / (phi(end-1) - phi(end-2));


%exclude startpoint and last endpoint as there isn't enough information to
%calculate them
aF([1, end]) = NaN;

aF = aF * w;

%-------------------velocity 0 points-------------------
% Initialize array to store indices of zero points
velocity_0 = [];

% Detect sign changes in vF array
for i = 1:length(phi)-1
   if (vF(i) < 0 && vF(i+1) > 0) || (vF(i) > 0 && vF(i+1) < 0)
      velocity_0(end+1) = i; % Store index of zero point
   end
end

% Convert zeros to corresponding phi values
velocity_0 = phi(velocity_0);

%====================== OUTPUT =================================
%plot the graph with all 3 functions
figure;
hold on
plot(phi, yF, 'black-', 'LineWidth', 0.8);
plot(phi, vF, 'b-', 'LineWidth', 0.8);
plot(phi, aF, 'r', 'LineWidth', 0.8);
plot(velocity_0, zeros(size(velocity_0)), 'ko', 'MarkerSize', 8);
hold off

axis ([0 6.5 -1 1]);
title ('Displacement, velocity and acceleration of slider F')
xlabel('Crank angle, rad');
ylabel('Displacement [m], velocity [m/s] and acceleration [m/s^2]');

grid on;

% Highlight points on the graph
legend('Displacement', 'Velocity', 'Acceleration', 'Location', 'SouthWest');


% Format x-axis ticks to show values in radians
xticks(0:pi/4:2*pi)
xticklabels({'0', '\pi/4','\pi/2', '3\pi/4', '\pi', '5\pi/4','3\pi/2', '7\pi/4','2\pi'})


% Output all calculated values

disp("The maximum height of Slider F (in m):");
disp(Maximum);
disp("The corresponding crank angle (in rad):");
disp(Maximum_x);
disp("The minimum height of slider F (in m):");
disp(Minimum);
disp("The corresponding crank angle (in rad):");
disp(Minimum_x);

disp("The points of zero velocity (in rad):")
disp(velocity_0)
```