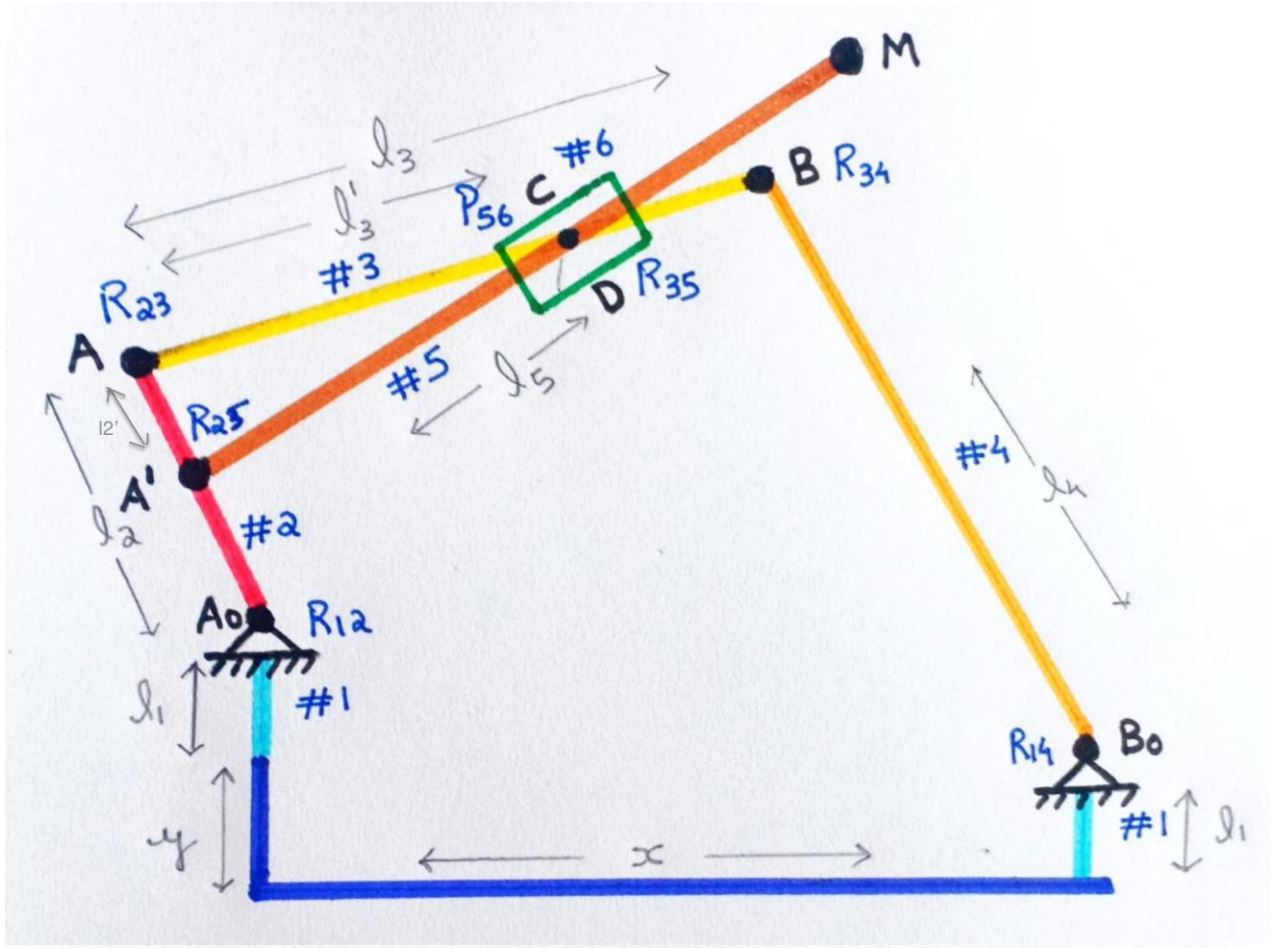
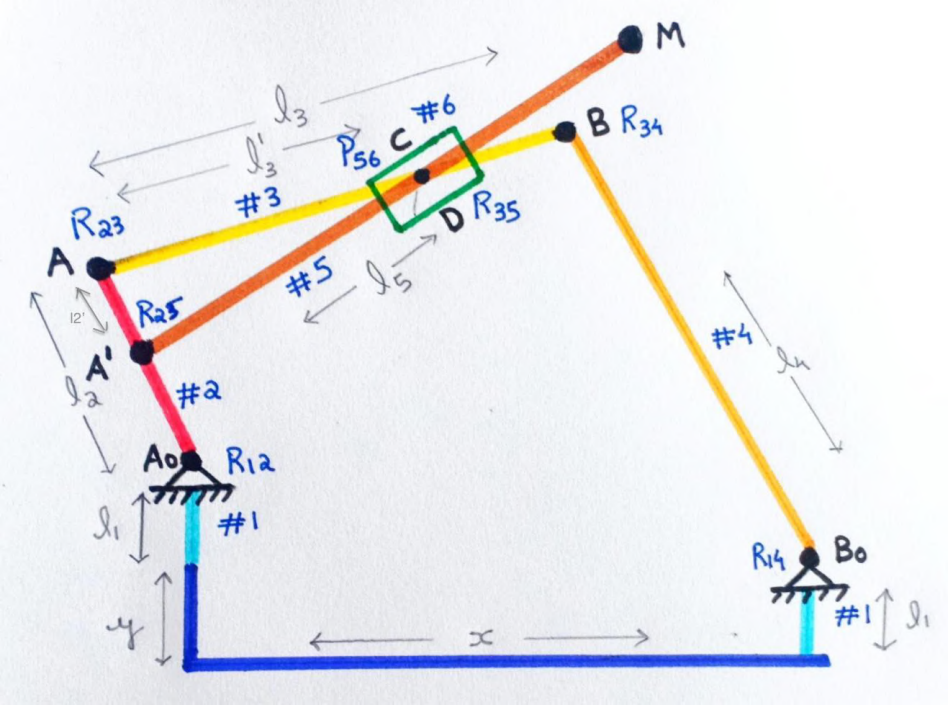
Question



- $x = 168\,mm$
- $y = 20\,mm$
- $l_2 = 20\,mm$
- $l_2' = -5\,mm$
- $l_3 = 100\,mm$
- $l_3' = 70\,mm$
- $l_4 = 100\,mm$
- $l_5 = 140\,mm$
- $l_1 = 10\,mm$

DOF



DOF=3(No. of moving links)-2(no. of lower pairs)-1(no. of higher pairs)

No. of moving links = 5;

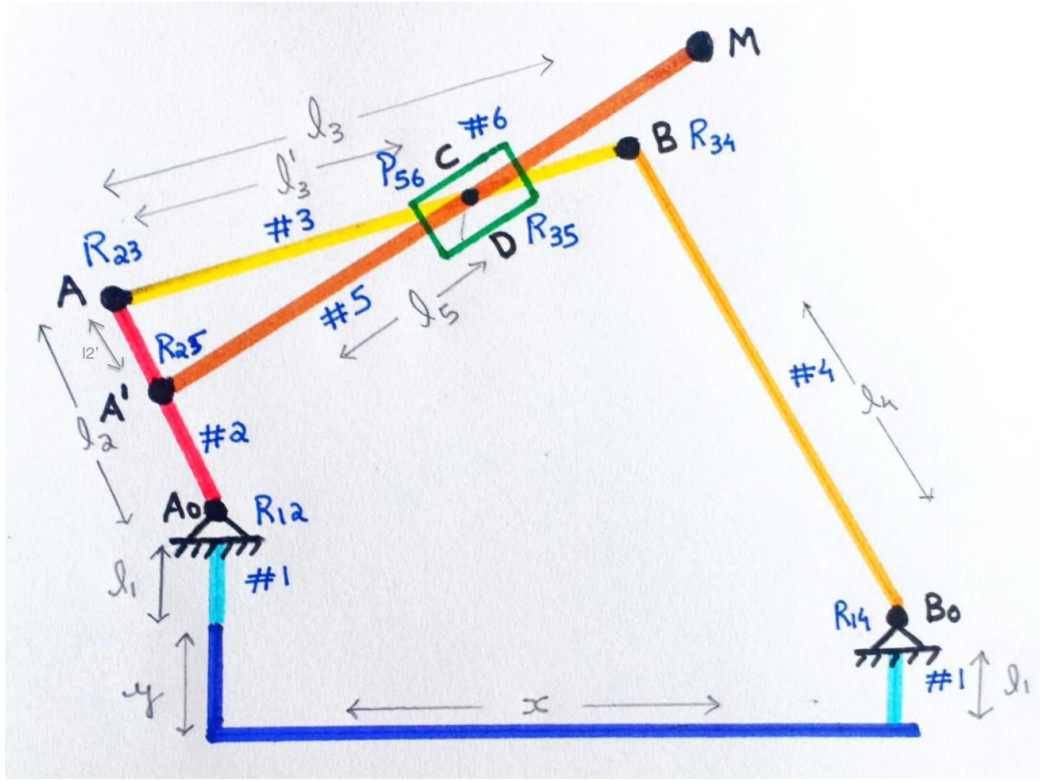
No. of Lower pairs = 6 revolute Joints + 1 prismatic pair = 7;

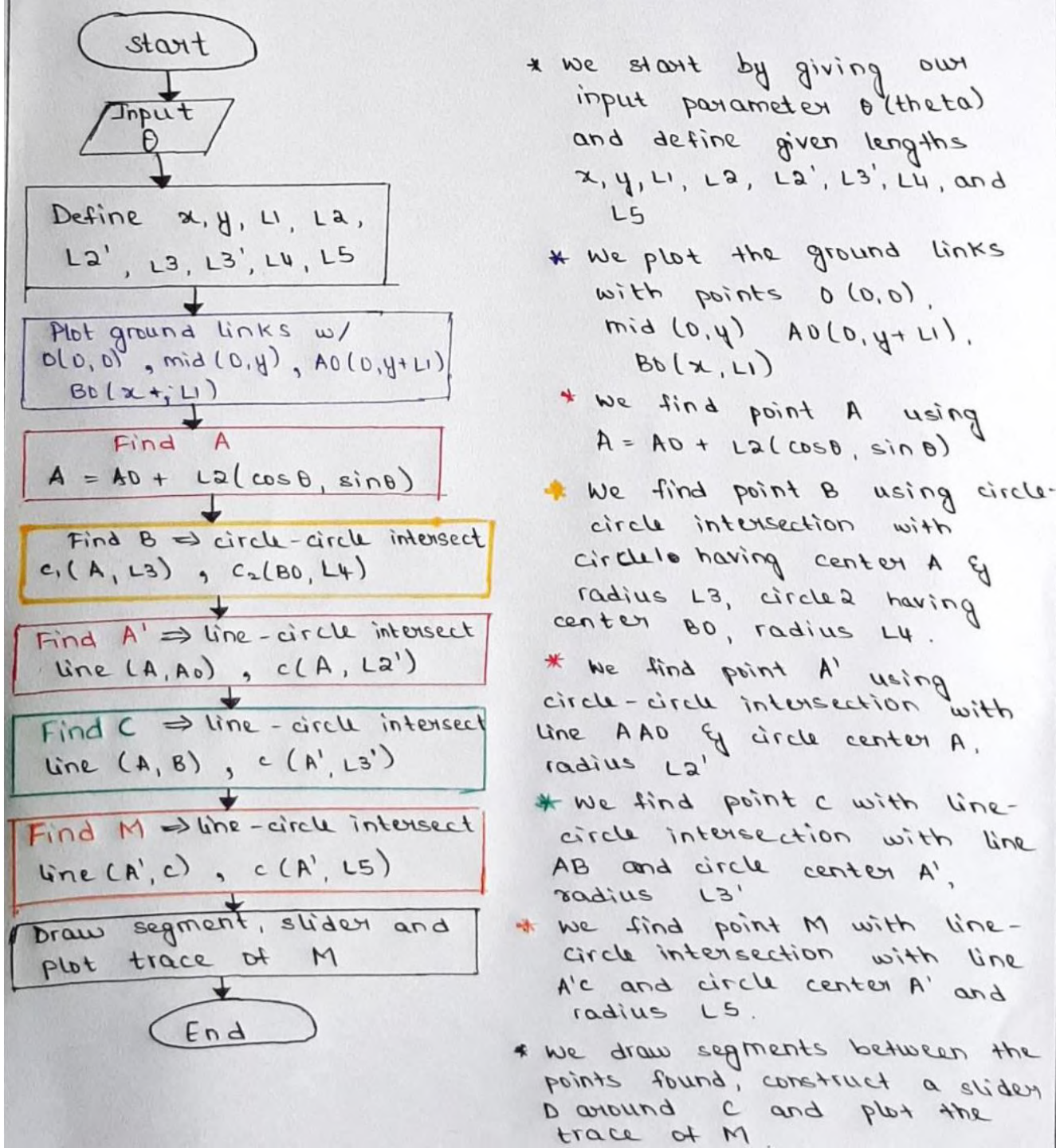
No. of higher pairs = 0;

DOF = 3(5)-2(7)-1(0) = **1**

The Degree of Freedom of the mechanism is 1, hence -considering that link dimensions are given- we require only one input parameter for the mechanism to function.

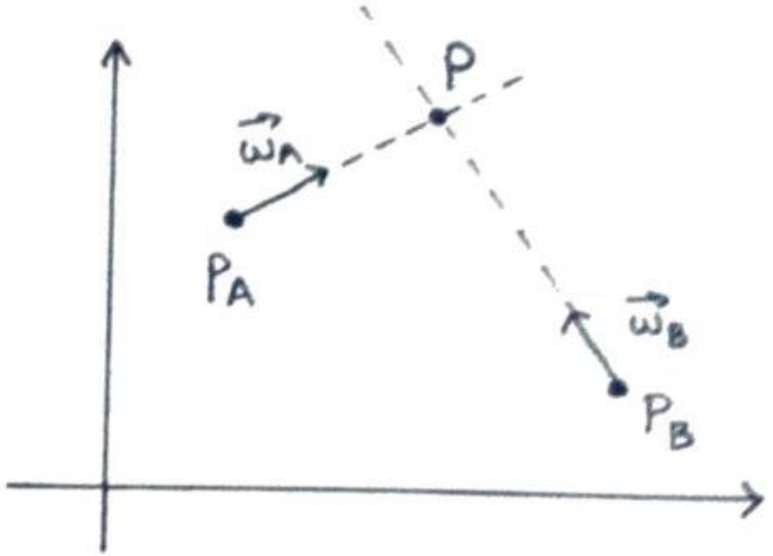
Position Analysis





* we start by giving our input parameter θ (theta) and define given lengths x, y, L1, L2, L2', L3', L4, and L5
* we plot the ground links with points O (0,0), mid (0,y) A0(0, y+L1), B0(x, L1)
* we find point A using A = A0 + L2(cosθ, sinθ)
* we find point B using circle-circle intersection with circle1 having center A & radius L3, circle2 having center B0, radius L4.
* we find point A' using circle-circle intersection with line AA0 & circle center A, radius L2'
* we find point C with line-circle intersection with line AB and circle center A', radius L3'
* we find point M with line-circle intersection with line A'C and circle center A' and radius L5.
* we draw segments between the points found, construct a slider D around C and plot the trace of M

Line-Line Intersection



Given $\vec{P}_A$, $\vec{P}_B$ & its unit direction $\vec{\omega}_A$ & $\vec{\omega}_B$ respectively
Find point of intersection $\vec{P}$

$$\vec{P} = \vec{P}_A + t_A \cdot \vec{\omega}_A \quad \text{—— ①}$$
$$\vec{P} = \vec{P}_B + t_B \cdot \vec{\omega}_B \quad \text{—— ②}$$

equate ① & ②

$$\begin{bmatrix} x_A \\ y_A \end{bmatrix} + t_A \begin{bmatrix} f_A \\ g_A \end{bmatrix} = \begin{bmatrix} x_B \\ y_B \end{bmatrix} + t_B \begin{bmatrix} f_B \\ g_B \end{bmatrix}$$

t_A & t_B are unknown parameters
hence solve for t_A, t_B & substitute in eqn ① or ②

Matrix form

$$\begin{bmatrix} f_a & -f_b \\ g_a & -g_b \end{bmatrix} \begin{bmatrix} t_A \\ t_B \end{bmatrix} = \begin{bmatrix} x_B - x_A \\ y_B - y_A \end{bmatrix}$$

by using cramer's rule

$$\det = f_B g_A - f_A g_B$$

if $\det = 0$, then lines are parallel or coincident.

$$t_A = \frac{\begin{vmatrix} (x_B - x_A) & -f_B \\ (y_B - y_A) & -g_B \end{vmatrix}}{(\det)}$$

$$\therefore t_A = \frac{f_B(y_B - y_A) - g_B(x_B - x_A)}{(\det)}$$

Illy $t_B = \dfrac{f_A(y_B - y_A) - g_A(x_B - x_A)}{(\det)}$

$\therefore$ intersection point $\vec{P}$

$$\vec{P} = P_A + t_A\vec{\omega}_A \quad \text{or} \quad \vec{P} = \vec{P}_B + t_B\vec{\omega}_B$$

Steps:

given - $\vec{P}_A$, $\vec{P}_B$, $\vec{\omega}_A$, $\vec{\omega}_B$

find - $\det \rightarrow t_A \rightarrow t_B \rightarrow \vec{P}$

Line-Circle Intersection

Given a line $\vec{P}_L$ with unit direction $\vec{w}_L$ and a circle with centre C and radius r.
find the point of intersection P_A & P_B

let $\vec{P}$ be a point on the line

$$\vec{P} = \vec{P}_L + t(\vec{w}_L) = \begin{bmatrix} x_L \\ y_L \end{bmatrix} + t\begin{bmatrix} f_L \\ g_L \end{bmatrix} \quad \text{—— (1)}$$

$\vec{P}$ on the circle

$$\vec{P} = \vec{C} + \begin{bmatrix} r\cos\theta \\ r\sin\theta \end{bmatrix} = \begin{bmatrix} x_c \\ y_c \end{bmatrix} + \begin{bmatrix} r\cos\theta \\ r\sin\theta \end{bmatrix} \quad \text{—— (2)}$$

equate (1) & (2)

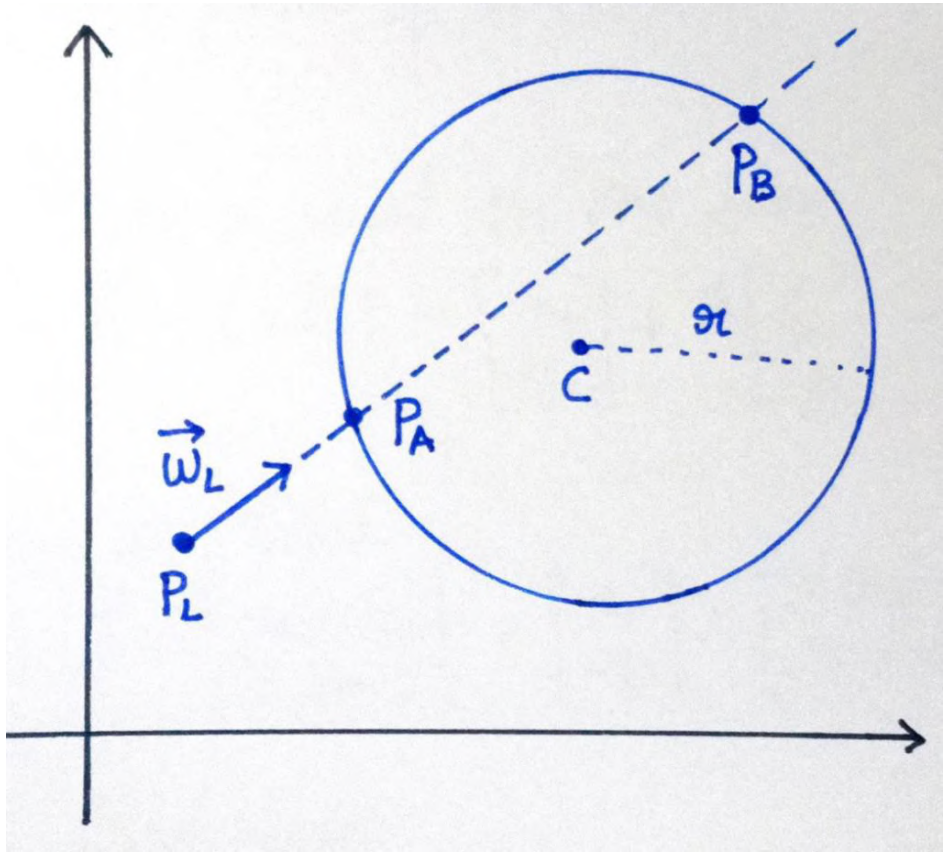
$$x_L + t f_L - x_c = r\cos\theta \quad \text{—— (3)}$$
$$y_L + t g_L - y_c = r\sin\theta \quad \text{—— (4)}$$

by squaring and adding (3) & (4) and after some manipulations and calculations.
we get a quadratic equation

$$at^2 + bt + c = 0$$

where

$$a = f_L^2 + g_L^2$$
$$b = 2(x_L - x_c)f_L + 2(y_L - y_c)g_L$$
$$c = (x_L - x_c)^2 + (y_L - y_c)^2 - r^2$$



$$\therefore \text{ Solution } = t = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$$t_B = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \qquad t_A = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

$$\therefore \quad P_A = P_L + t_A W_L \quad - 1^{st} \text{ intersection}$$
$$P_B = P_L + t_B W_L \quad - 2^{nd} \text{ intersection}$$

if $b^2 - 4ac > 0 \implies$ 2 roots

$b^2 - 4ac = 0 \implies$ 1 root

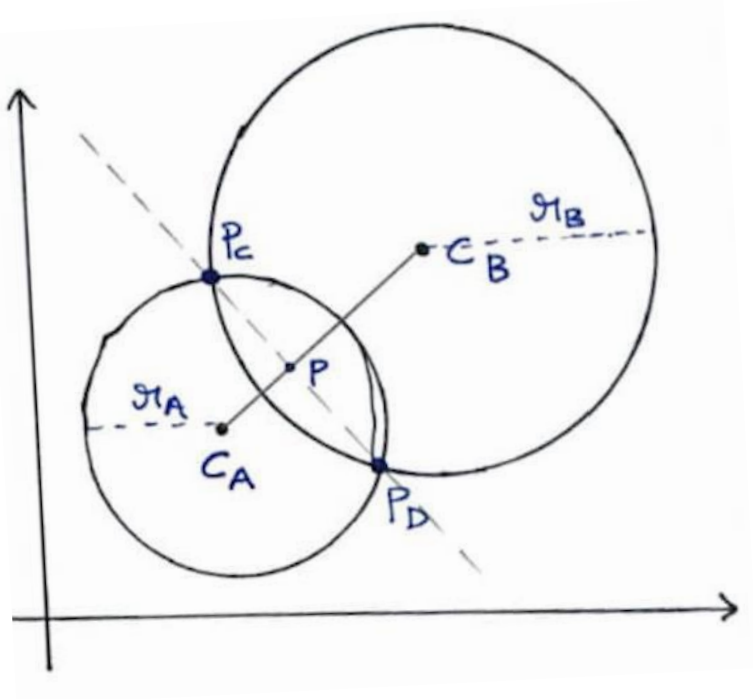
$b^2 - 4ac < 0 \implies$ no root

MATLAB Code for LineCircle Intersection

LineCircleIntersection.m

```
function [pointIntersectionA, pointIntersectionB] = LineCircleIntersection(lineStart, lineEnd, circleCenter, r)
pointIntersectionA = []; pointIntersectionB = []; %empty arrays would be returned if any error
deltaLVector = lineEnd - lineStart; lengthLVector = norm(deltaLVector);
dirLine = deltaLVector/lengthLVector; %Normalize as unit vector
%defining simpler variables to match with derivation
xL = lineStart(1); yL = lineStart(2); fL = dirLine(1);   gL = dirLine(2);
xC = circleCenter(1); yC = circleCenter(2);
denomTerm = fL*fL + gL*gL;
tempTerm = fL*(yL-yC)-gL*(xL-xC);
discriminant = r*r*denomTerm - tempTerm*tempTerm;
    if discriminant < 0
      return; %Non intersecting case
    end
    if discriminant == 0
      t = (fL*(xC-xL) + gL*(yC-yL))/denomTerm;
        tA = t; tB = t; %equal and
    else
        tExpression1 = (fL*(xC-xL)+gL*(yC-yL))/denomTerm;
     tExpression2 = sqrt(r*r*denomTerm - (fL*(yL-yC)-gL*(xL-xC))*(fL*(yL-yC)-gL*(xL-xC)));
     tA = tExpression1 + tExpression2;
     tB = tExpression1 - tExpression2;
    end

    if tA >= 0 && tA <= lengthLVector
     pointIntersectionA = lineStart + tA*dirLine;
    end

    if tB >= 0 && tB <= lengthLVector
     pointIntersectionB = lineStart + tB*dirLine;
    end

end

```

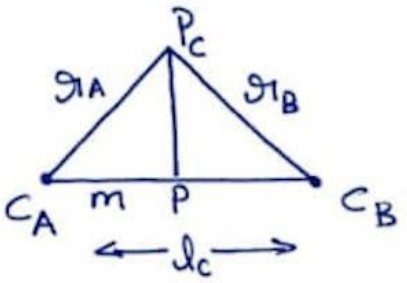
Circle-Circle Intersection



Given 2 circles with centre C_A, C_B and radius r_A, r_B respectively.

- find intersection points P_c, P_D.

$C_AC_B \perp P_cP_D$



$m = C_AP$

$l_c = C_AC_B$

$l_c - m = PC_B$

$r_A^2 = m^2 + (P_cP)^2$ —— ①

$r_B^2 = (l_c - m)^2 + (P_cP)^2$ —— ②

from ① & ②

$$m = \frac{r_A^2 - r_B^2 + l_c^2}{2 \cdot l_c}$$

unit direction along $C_AC_B = \vec{w}_{AB} = \dfrac{(\vec{C}_B - \vec{C}_A)}{|\vec{C}_B - \vec{C}_A|}$

$$\therefore \vec{P} = \vec{C}_A + m \cdot \vec{w}_{AB}$$

$\vec{w}_{DC} \perp$ to $\vec{w}_{AB}$

$$\therefore \vec{w}_{DC} = \begin{bmatrix} -q_{AB} \\ f_{AB} \end{bmatrix}$$

$$w_{AB} = \begin{bmatrix} f_{AB} \\ q_{AB} \end{bmatrix}$$

To find P_c, we have line through P & its unit direction ω_{DC}

$P_cP^2 = r_A^2 - m^2$

$P_cP = h = \sqrt{r_A^2 - m^2}$ if h - +ve $\Rightarrow$ 2 unique points of intersection

h - -ve $\Rightarrow$ not intersecting

h - 0 $\Rightarrow$ tangential (1 Point)

$$\vec{P_c} = \vec{P} + h(\vec{\omega}_{DC})$$
$$\vec{P_D} = \vec{P} - h(\vec{\omega}_{DC})$$

Steps to be followed:

i/p - C_A, C_B, r_A, r_B

find $\vec{\omega}_{AB} \rightarrow \vec{\omega}_{DC} \rightarrow m \rightarrow \vec{P} \rightarrow h \rightarrow \vec{P_c}$ & $\vec{P_D}$

points of intersection are $\vec{P_c}$ & $\vec{P_D}$

MATLAB Code for CircleCircle Intersection

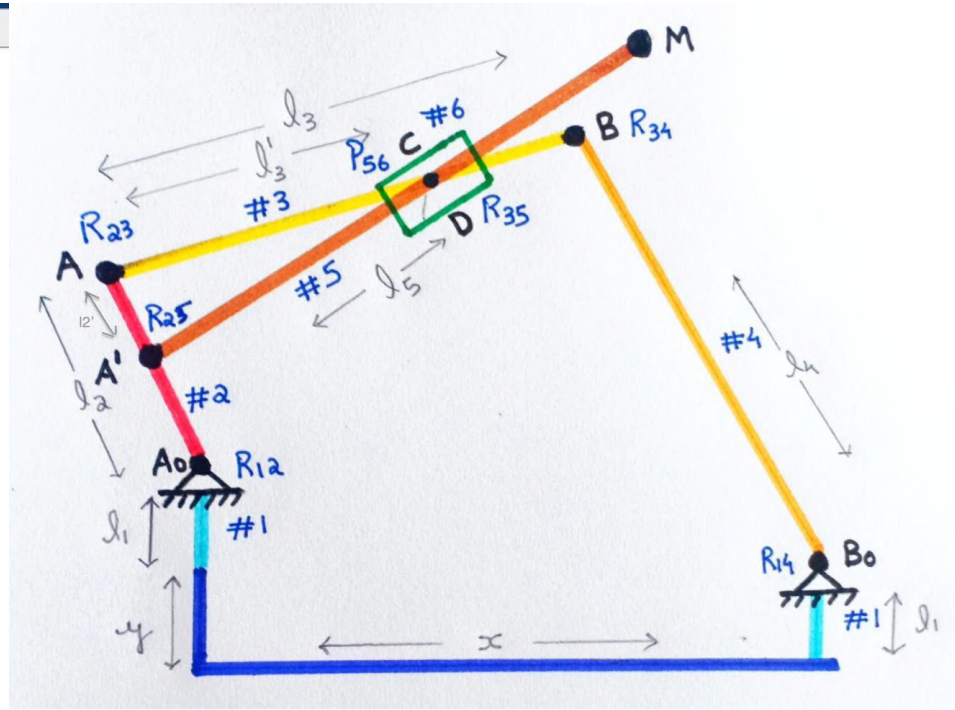
CircleCircleIntersection.m*

```
function [pointIntersectionA, pointIntersectionB] = CircleCircleIntersection(circleCenterA, rA, circleCenterB, rB)
pointIntersectionA = []; pointIntersectionB = []; %empty arrays would be returned if any error
deltaCVector = circleCenterB - circleCenterA; lengthCVector = norm(deltaCVector);
dirCVector = deltaCVector/lengthCVector; %Normalize as unit vector
%defining simpler variables to match with derivation
xCA = circleCenterA(1); yCA = circleCenterA(2);
m=(lengthCVector*lengthCVector + rA*rA - rB*rB)/(2*lengthCVector);

chordCenterPoint = circleCenterA + m*(dirCVector);
dirLine = [-dirCVector(2); dirCVector(1)]; %Rotation about Z axis by 90degree => dirChord = dirCVector*[0 1; -1 0] (row vector!)

xL = chordCenterPoint(1); yL = chordCenterPoint(2); fL = dirLine(1); gL = dirLine(2);

denomTerm = fL*fL + gL*gL;
tempTerm = fL*(yL-yCA)-gL*(xL-xCA);
discriminant = rA*rA*denomTerm - tempTerm*tempTerm;
if discriminant < 0
    return; %Non intersecting case, empty arrays are not populated
end

if discriminant == 0
    t = (fL*(xCA-xL) + gL*(yCA-yL))/denomTerm;
    tA = t; tB = t; %equal and real
else
    tExpression1 = (fL*(xCA-xL)+gL*(yCA-yL))/denomTerm;
    tExpression2 = sqrt(rA*rA*denomTerm - (fL*(yL-yCA)-gL*(xL-xCA))*(fL*(yL-yCA)-gL*(xL-xCA)));
    tA = tExpression1 + tExpression2;
    tB = tExpression1 - tExpression2;
end

pointIntersectionA = chordCenterPoint + tA*dirLine;
pointIntersectionB = chordCenterPoint + tB*dirLine;

end
```

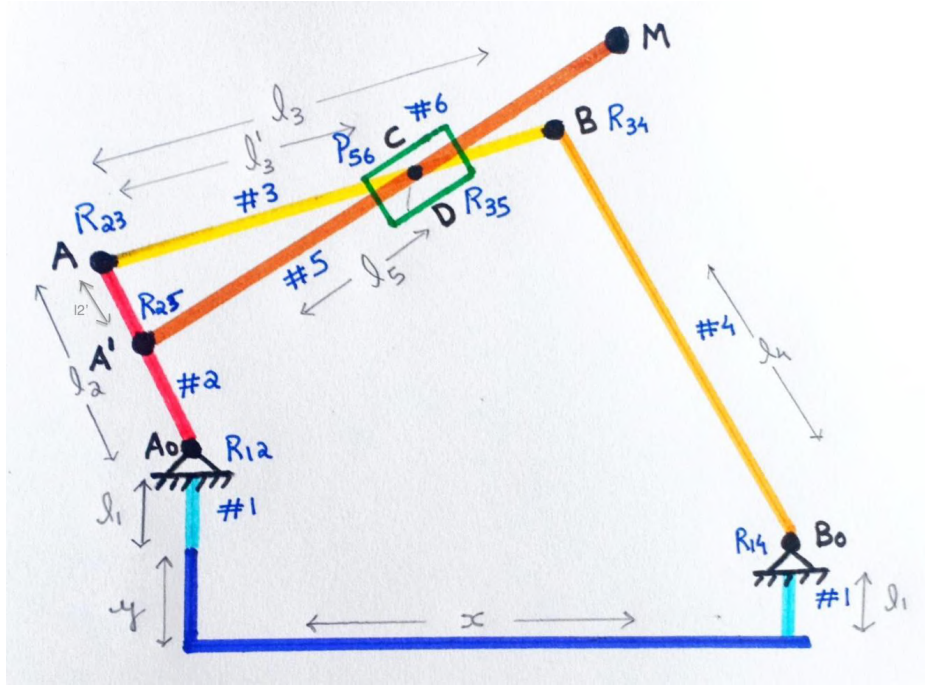
MATLAB Program: Mechanism Animation

endsem6bar.m

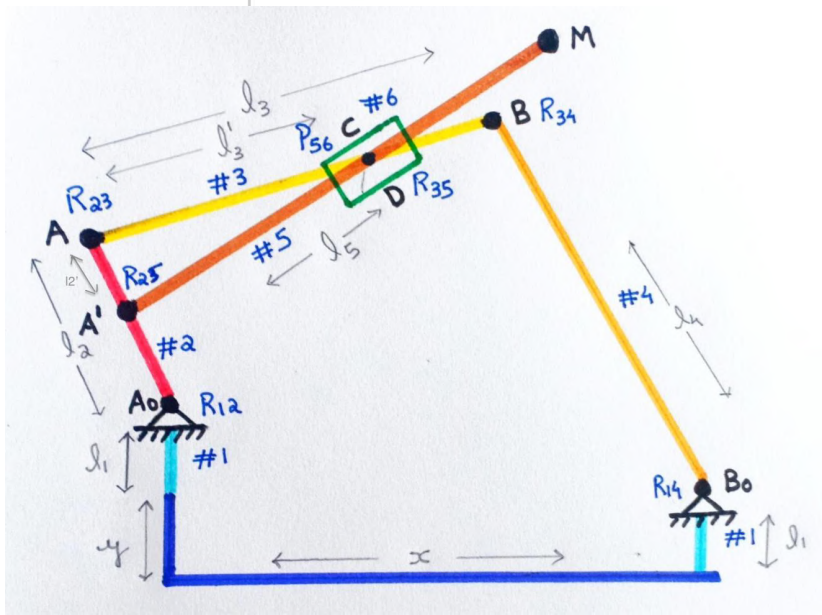
```
clc;clf;clearvars;
% Defining the length of the links
fx=168; fy=20; L2=20; L2dash=5; L3=100;
L3dash=70; L4=100; L5=140; L1=10;

% Defining the fixed points
pointO = [0; 0];
pointB0 = [fx; L1];
pointA0 = [0; fy+L1];
pointmid = [0;fy];
pointOdash=[fx;0];

% DOF = Theta
thetaDegreesArray = 0:16:1800; % Theta in degrees
thetaRadiansArray = thetaDegreesArray*(pi/180.0);% Theta in radians

thetaInitial = thetaRadiansArray(1);
% Defining Point A
pointA = pointA0 + L2*[cos(thetaInitial); sin(thetaInitial)];
[pointB1, pointB2] = CircleCircleIntersection(pointA, L3, pointB0, L4);

% Choosing One of the points of intersection of the 2 circles to be Point B
pointB = pointB1;

% Point A is a point of Intersection of lineA0A &
% Circle with centre A and  radius L2dash
[pointAdash1, pointAdash2] = LineCircleIntersection(pointA0, pointA, pointA, L2dash);
 pointAdash = pointAdash2;

% Point C is a point of Intersection of lineAB &
% Circle with centre A and  radius L3dash
[pointC1, pointC2] = LineCircleIntersection(pointA, pointB, pointAdash, L3dash);
 pointC = pointC1;


 % Defining Point M
 [pointM1, pointM2] = RayCircleIntersection(pointAdash, pointC, pointAdash, L5);
 pointM = pointM2;
```



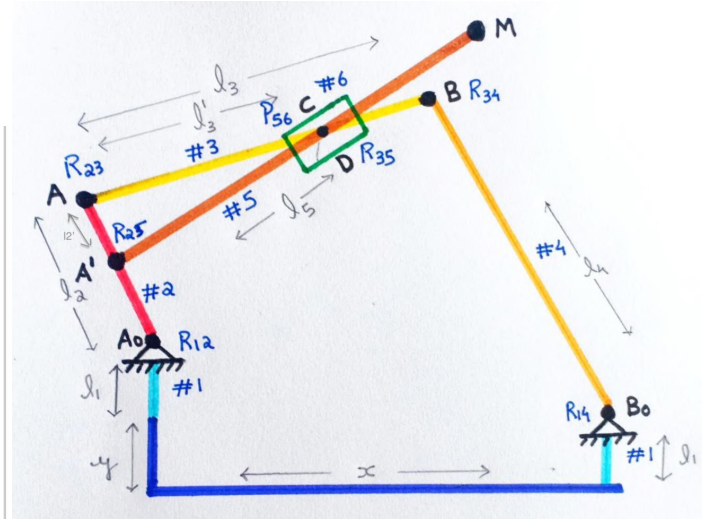
```
39
40      %hold on
41
42      %Animation
43 -    figure(1)
44 -    grid on,set(gca,'FontSize',15)
45 -    set(gcf,'Position',[100 100 1200 900])
46 -    grid on
47
48      % Creating Zero arrays of x&y Co-ordinates of Point M
49 -    Mxarray=zeros(length(thetaRadiansArray));
50 -    Myarray=zeros(length(thetaRadiansArray));
51
52 -    for index = 1:length(thetaRadiansArray)
53 -        theta = thetaRadiansArray(index);
54
55 -        pointA = pointA0 + L2*[cos(theta); sin(theta)];
56
57 -        [pointB1, pointB2] = CircleCircleIntersection(pointA, L3, pointB0, L4);
58 -        pointB = pointB1;
59
60 -        distBetweenPrevBandB1 = norm(pointB-pointB1);
61 -        distBetweenPrevBandB2 = norm(pointB-pointB2);
62
63 -        [pointAdash1, pointAdash2] = LineCircleIntersection(pointA0, pointA, pointA, L2dash);
64 -         pointAdash = pointAdash2;
65
66 -        [pointC1, pointC2] = LineCircleIntersection(pointA, pointB, pointAdash, L3dash);
67 -        pointC = pointC1;
68
69
70 -        [pointM1, pointM2] = RayCircleIntersection(pointAdash, pointC, pointAdash, L5);
71 -        pointM = pointM2;
72          % Populating the x&y Co-ordinate arrays of Point M
73 -        Mxarray(index)=pointM(1);
74 -        Myarray(index)=pointM(2);
```



```
75
76      % Unit Direction of line A'C
77 -    w=(pointC-pointAdash)/norm(pointC-pointAdash);
78      % Angle of inclination of line A'C
79 -    thetasliding=atan2(w(2),w(1));
80
81      % Definig the points of the Slider wrt PointC by using angle of inclination
82      % of line A'c
83 -    m1 = [pointC(1)+10*cos(thetasliding+(pi/4));pointC(2)+10*sin(thetasliding+(pi/4))];
84 -    m2 = [pointC(1)+10*cos(thetasliding+(3*pi/4));pointC(2)+10*sin(thetasliding+(3*pi/4))];
85 -    m3 = [pointC(1)+10*cos(thetasliding-(3*pi/4));pointC(2)+10*sin(thetasliding-(3*pi/4))];
86 -    m4 = [pointC(1)+10*cos(thetasliding-(pi/4));pointC(2)+10*sin(thetasliding-(pi/4))];
87
88
89 -    plot([pointO(1) pointmid(1)], [pointO(2) pointmid(2)], 'b','LineWidth',5);% y
90 -    hold on;
91      % Plotting all the points
92 -    plot(pointAdash(1),pointAdash(2),'k.','MarkerSize',20)
93 -    plot(pointM(1),pointM(2),'k.','MarkerSize',30)
94 -    plot(Mxarray,Myarray,'k.','MarkerSize',5) % Trace
95 -    plot(pointC(1),pointC(2),'g.','MarkerSize',20)
96 -    plot(pointB(1),pointB(2),'k.','MarkerSize',30)
97 -    plot(pointA0(1),pointA0(2),'k.','MarkerSize',30)
98 -    plot(pointB0(1),pointB0(2),'k.','MarkerSize',30)
99 -    plot(pointA(1),pointA(2),'k.','MarkerSize',20)
```

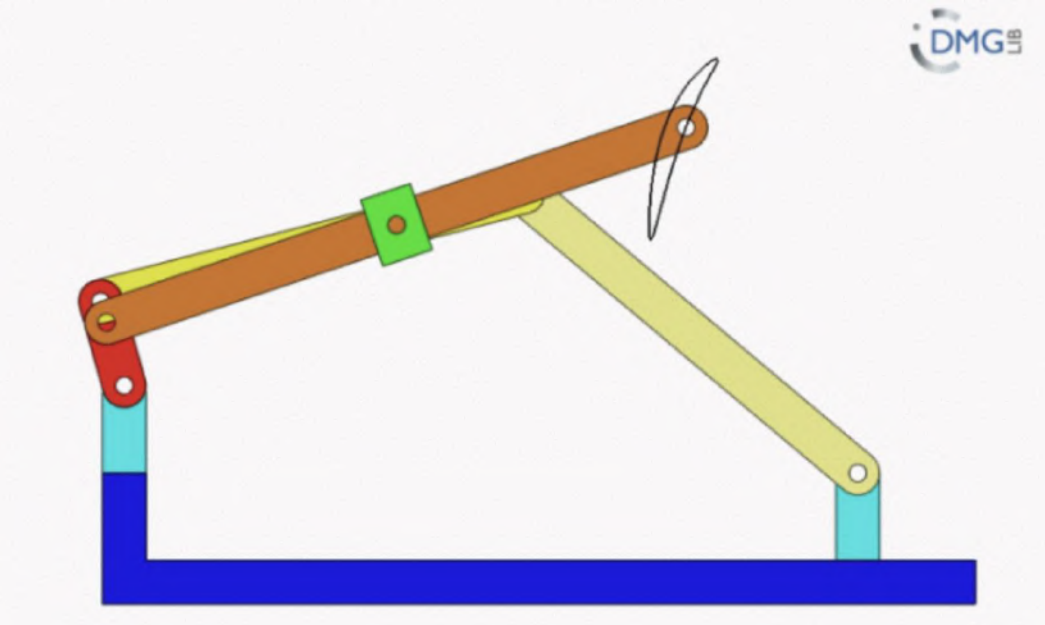


```
100
101         % Defining Colors
102 -       Orange = '#ffb76a';
103 -       OrangeColor = sscanf(Orange(2:end),'%2x%2x%2x',[1 3])/255;
104 -       yellowBright = '#fbfe32';
105 -       yellowBrightColor = sscanf(yellowBright(2:end),'%2x%2x%2x',[1 3])/255;
106 -       yellowDull = '#f3f700';
107 -       yellowDullColor = sscanf(yellowDull(2:end),'%2x%2x%2x',[1 3])/255;
108
109         % Plotting all the links
110 -       plot([pointAdash(1) pointM(1)], [pointAdash(2) pointM(2)],'color',OrangeColor,'LineWidth',3.5);%L1
111 -       plot([pointmid(1) pointA0(1)], [pointmid(2) pointA0(2)], 'c','LineWidth',3.5);%L1
112 -       plot([pointA0(1) pointA(1)], [pointA0(2) pointA(2)], 'r-','LineWidth',3.5);%L2
113 -       plot([pointA(1) pointB(1)], [pointA(2) pointB(2)],'color',yellowBrightColor,'LineWidth',3.5);%L3
114 -       plot([pointB(1) pointB0(1)], [pointB(2) pointB0(2)],'color',yellowDullColor,'LineWidth',3.5);%L4
115 -       plot([pointB0(1) pointOdash(1)], [pointB0(2) pointOdash(2)], 'c-','LineWidth',3.5);%L1
116 -       plot([pointOdash(1) pointO(1)], [pointOdash(2) pointO(2)], 'b-','LineWidth',5);%x
117 -       title('Team-5  (InvertedSliderCrank connected in parallel to a FourBarMech)', 'FontSize', 24);
118 -       xlabel('X-axis', 'FontSize', 24);
119 -       ylabel('Y-axis', 'FontSize', 24)
120
121         % Plotting the slider
122 -       plot([m1(1) m2(1) m3(1) m4(1) m1(1)], [m1(2) m2(2) m3(2) m4(2) m1(2)],'g','LineWidth',3);hold on
123 -       axis([-40 200 -60 140]); % can be set as per the link dimensions
124 -       hold off;
125 -       drawnow();
126 -   end
127
```



MATLAB Program: Mechanism Screenshot

Question



Animation