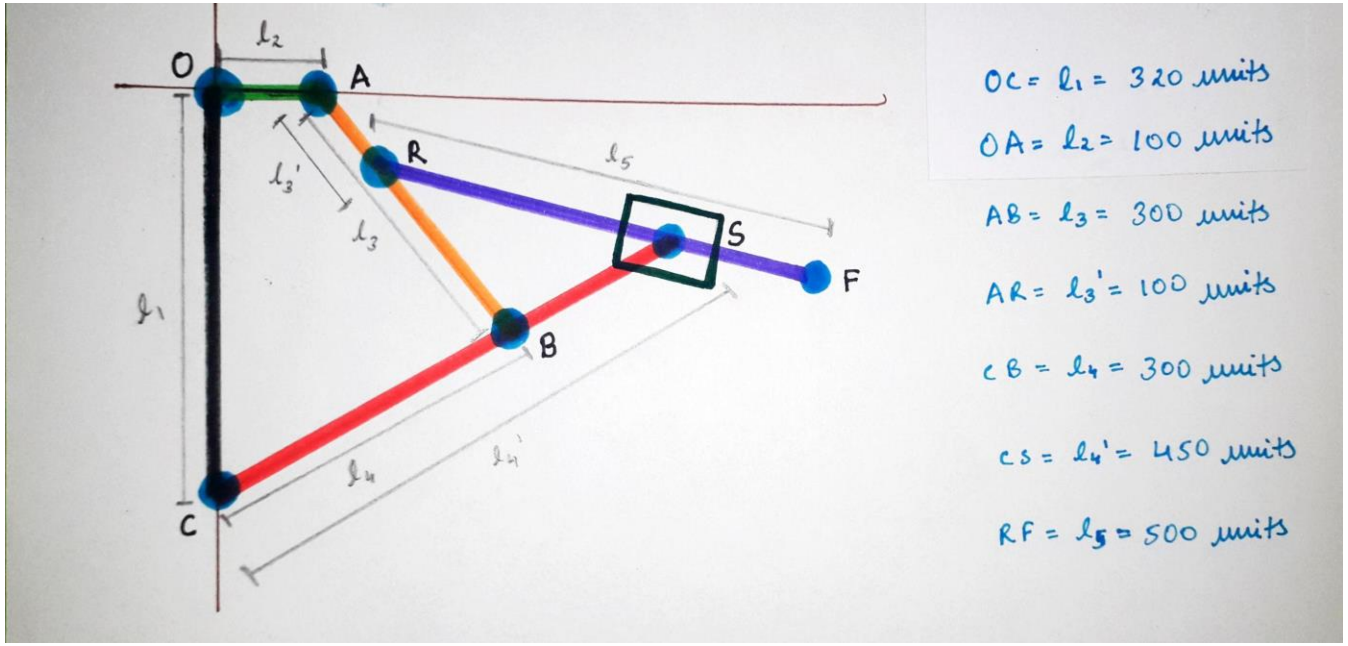
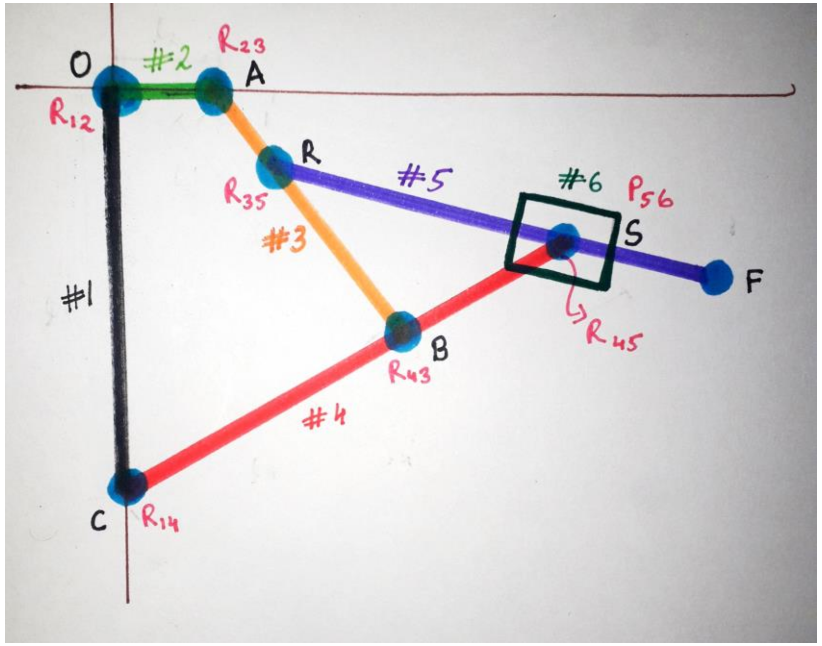
Question


O
A
R
S
F
B
C
l_2
l_3'
l_3
l_5
l_1
l_4
l_4'
$OC = l_1 = 320$ units
$OA = l_2 = 100$ units
$AB = l_3 = 300$ units
$AR = l_3' = 100$ units
$CB = l_4 = 300$ units
$CS = l_4' = 450$ units
$RF = l_5 = 500$ units


O
A
R
S
F
B
C
#1
#2
#3
#4
#5
#6
R_{12}
R_{23}
R_{35}
P_{56}
R_{45}
R_{43}
R_{14}

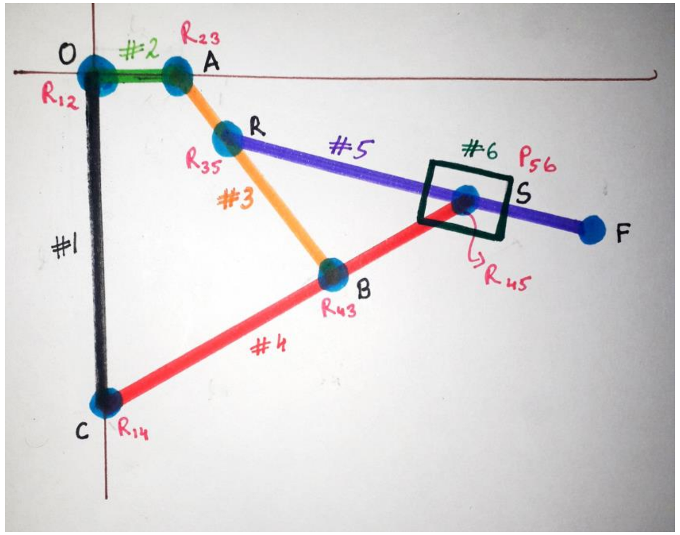
DOF

Degree of freedom is the number of independent parameters to represent the system uniquely.

Kutzbach Equation to find DOF:
DOF = 3(Number of links-1) – 2(Number of lower pairs) – 1(Number of higher pairs)

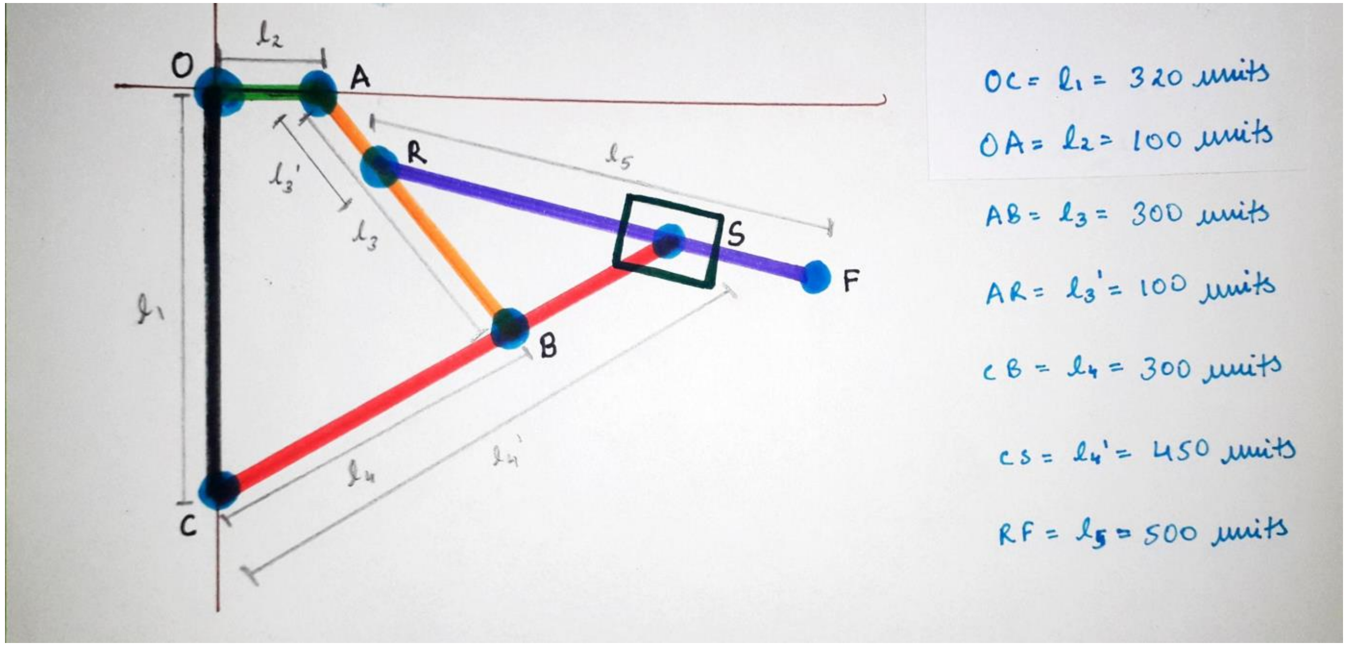
Here, Number of links = 6 (#1, #2, #3, #4, #5, #6)
Number of lower pairs = 7 (R_{12}, R_{23}, R_{35}, R_{14}, R_{43}, R_{45}, P_{56})
Number of higher pairs = 0

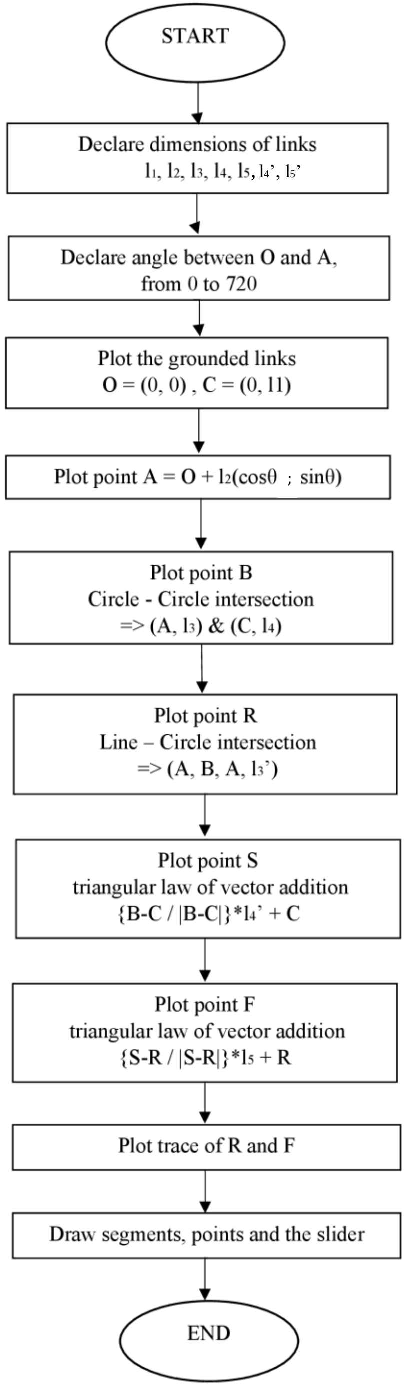
(P - Prismatic joints deliver a linear motion along the axis
R - Rotary joint uses a rotational motion along the joint axis)



DOF = 3(6-1) – 2(7) – 1(0)
= 3(5) – 2(7)
= 15 – 14
DOF = 1 (Correct, as we require only one input parameter, which is theta)

Position Analysis





Position analysis steps:

- Declare all the lengths, l_1, l_2, l_3, l_3’, l_4, l_4’ ,l_5
- Declare the angle theta between O and A from 0 to 720 in steps of 5
- Plot the grounded links, O = (0, 0), C = (0, l_1)
- Plot point A = O + l_2(cosθ + sinθ) { Point = point + r.(cosθ ; sinθ) }
- Plot point B, Circle-Circle Intersection => (ciecleCenterA, radiusA, circleCenterB, radiusB) => (A, l_3, C, l_4)
- Plot point R, Line-Circle Intersection => (lineStart, lineEnd, circleCenter, radius) => (A, B, A, l_3’)
- Plot point S, Triangular law of vector addition,

  (here, in triangle OSC, CB and CS have same direction) => {B-C / |B-C|}*l_4’ + C
- Plot point F, Triangular law of vector addition,

  (here, in triangle ORF, RF and RS have same direction) => {S-R / |S-R|}*l_5 + R
- Plot the trace of point R and point F
- Draw segments (OC, OA, AB, BC, CS, RF), points(O, A, R, B, C, S) and the slider

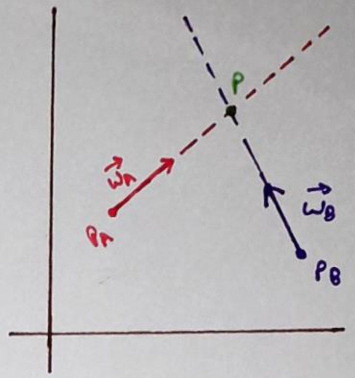
Line – Line Intersection

Here,

P is the intersection point

$L_A \rightarrow$ line A passing through P_A and having $\vec{w}_A$ as unit vector direction.

$L_B \rightarrow$ line B passing through P_B and having $\vec{w}_B$ as unit vector direction.



$$\vec{P} = \vec{P}_A + t_A \vec{w}_A \quad \text{—— (1)}$$

$$\vec{P} = \vec{P}_B + t_B \vec{w}_B \quad \text{—— (2)}$$

equate (1) and (2)

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

unknown parameters $\rightarrow$ t_A & t_B

$$\therefore \begin{bmatrix} f_A & -f_B \\ g_A & -g_B \end{bmatrix} \begin{bmatrix} t_A \\ t_B \end{bmatrix} = \begin{bmatrix} x_B & -x_A \\ y_B & -y_A \end{bmatrix} \qquad \left\{ \begin{array}{l} Ax = \vec{b} \\ x = A^{-1}\vec{b} \end{array} \right\}$$

Solving using Cramers ~~method~~ rule

$$\det = \begin{vmatrix} f_A & -f_B \\ g_A & -g_B \end{vmatrix}$$

{check if det = 0, if so, then lines are parallel or coincident}

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

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

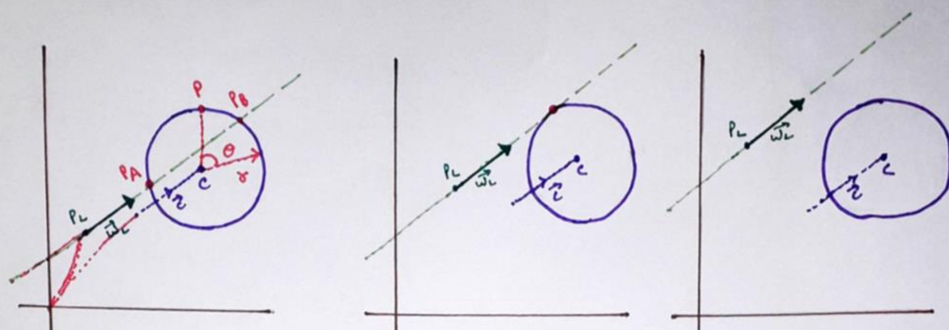
$$= \frac{f_B(y_B - y_A) - g_B(x_B - x_A)}{\det} \qquad = \frac{f_A(y_B - y_A) - g_A(x_B - x_A)}{\det}$$

$\therefore$ Intersecting points are,

$$\vec{P} = \vec{P_A} + t_A \vec{\omega_A}$$

$$\vec{P} = \vec{P_B} + t_B \vec{\omega_B}$$

Line – Circle Intersection



Line as a chord (2 intersections)   Line as a Tangent (1 intersection)   No intersections

Here, P_L = point on the line

$\vec{w_L}$ = unit vector along the line

$\vec{c}$ = center of the circle

r = radius of the circle

P_A and P_B are intersection points

$\vec{P}$ 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,

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

Intersection point ⇒ ① = ② } → 2 unknowns, θ & t

We square both sides and add them, we get a quadratic equation in t and θ parameter has been eliminated.

$$At^2 + Bt + C = 0$$

where,

$$A = f_c^2 + g_c^2$$

$$B = 2(x_L - x_c)f_L + 2(y_L - y_c)g_L$$

$$C = (x_L - x_c)^2 + (y_L - y_c)^2 - r^2$$

Solution to the quadratic equation

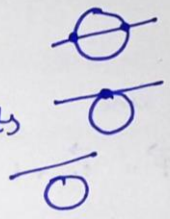
$$t = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A}$$

if discriminant, $\Delta > 0$, real & unequal roots

$\Delta = B^2 - 4AC$

$\Delta = 0$, real & unrepeated roots

$\Delta < 0$, complex roots

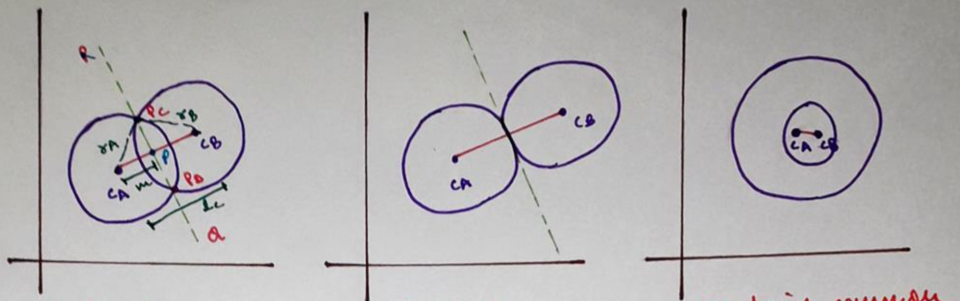


$\therefore$ Intersection points are,

$$\vec{P_A} = \vec{P_L} + t_A \vec{\omega_L}$$

$$\vec{P_B} = \vec{P_L} + t_B \vec{\omega_L}$$

Circle – Circle Intersection



2 point in common   1 point in common   NO point common

Here,

P_cP_D is the common chord

P is the midpoint of C_AC_B line segment if radii are equal

ΔC_APP_c, $r_A^2 = m^2 + (P_cP)^2$ —— ①

ΔC_BPP_c, $r_B^2 = (l_c - m)^2 + (P_cP)^2$ —— ②

from ① and ②,

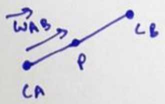
$$m = \frac{r_A^2 - r_B^2 + l_c^2}{2l_c}$$

$m = C_AP$

$l_c = C_AC_B$

$l_c - m = PC_B$

Now that we know m, point P can be found



$$\vec{\omega}_{AB} = \frac{\vec{C_B} - \vec{C_A}}{|\vec{C_B} - \vec{C_A}|}$$

$$\therefore \vec{P} = \vec{C_A} + m\,\vec{\omega}_{AB}$$

We know that $\vec{w}_{DC}$ is perpendicular to $\vec{w}_{AB}$

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

P_c passes through P and is in the direction of $\vec{w}_{DC}$.

$$\therefore (PP_c)^2 = r_A^2 - m^2$$

$$P_c = h = \sqrt{r_A^2 - m^2}$$

if $h = 0$, it is a tangent

$h > 0$, has 2 distinct points

$h < 0$, non-intersecting

$\therefore$ The intersecting points are,

$$\vec{P_c} = \vec{P} + h(\vec{w}_{DC})$$

$$\vec{P_D} = \vec{P} - h(\vec{w}_{DC})$$

MATLAB Program

```
clc;
clear all;
clf;
```

Declaring all the lengths of the links

```
l1 = 320;
l2 = 100;
l3 = 300;
l3_1 = 100;
l4 = 300;
l4_1 = 450;
l5 = 500;
```

DOF = theta (one input parameter)

Giving the value of theta from 0 to 360 degrees to move in steps of 5 and later converting theta from degree to radians

```
thetaDegreesArray = 0:5:720;
thetaRadiansArray = thetaDegreesArray*(pi/180.0);
thetaInitial = thetaRadiansArray(1);
```

Delaring the known points, here point O and point C which are the grounded points

```
pointO = [0; 0];
pointC = [0 ; -l1];
```

Finding point A

```
pointA = pointO + l2*[cos(thetaInitial); sin(thetaInitial)];
```

Finding point B using Circle - Circle Intersection

```
[pointB1, pointB2] = CircleCircleIntersection(pointA, l3, pointC, l4);
```

Choosing one of the solutions as the branch of point B.

One is to the left and one is to the right, here, we use the one to the right which is point B1

```
pointB = pointB1;
```

Finding point R using Line - Circle Intersection

```
[pointR1, pointR2] = LineCircleIntersection(pointA, pointB, pointA, l3_1);
```

Choosing one of the solutions as the branch of point R

```
pointR = pointR1;
```

Finding point S using Triangular law of vector addition,

```
pointS = (((pointB - pointC)/(norm(pointB - pointC)))*l4_1) + pointC;
```

Finding point F using Triangular law of vector addition,

```
pointF = (((pointS - pointR)/(norm(pointS - pointR)))*l5) + pointR;
```

Code for Animation

```
for index = 1:length(thetaRadiansArray)
    theta = thetaRadiansArray(index);

    pointA = pointO + l2*[cos(theta); sin(theta)];

    [pointB1, pointB2] = CircleCircleIntersection(pointA, l3, pointC, l4);
    distBetweenPrevBandB1 = norm(pointB-pointB1);
    distBetweenPrevBandB2 = norm(pointB-pointB2);
    %Choose the solution that is nearest to the previous point B
    if(distBetweenPrevBandB1 < distBetweenPrevBandB2)
        pointB = pointB1;
    else
        pointB = pointB2;
    end

    [pointR, pointR2] = LineCircleIntersection(pointA, pointB, pointA, l3_1);
    pointR = [pointR(1) ; pointR(2)];

    pointS = (((pointB - pointC)/(norm(pointB - pointC)))*l4_1) + pointC;
```

```
    pointF = (((pointS - pointR)/(norm(pointS - pointR)))*15) + pointR;
```

Code for trace of Point R

```
    couplerMidpointTraceXArray(index) = pointR(1);
    couplerMidpointTraceYArray(index) = pointR(2);
```

Code for trace of Point F

```
    couplerMidpointTraceXArray1(index) = pointF(1);
    couplerMidpointTraceYArray1(index) = pointF(2);
    hold off;
```

Draw the axis in Brown color

```
    Brown = '#964B00';
    BrownColor = sscanf(Brown(2:end),'%2x%2x%2x',[1 3])/255;

    plot([-141 564], [0 0], 'color', BrownColor, 'LineWidth', 1)
    hold on
    plot([0 0], [-350 160], 'color', BrownColor, 'LineWidth', 1)
    hold on
```

Plot all the lines

```
    Orange = '#FF6700';
    OrangeColor = sscanf(Orange(2:end),'%2x%2x%2x',[1 3])/255;

    Purple = '#800080';
    PurpleColor = sscanf(Purple(2:end),'%2x%2x%2x',[1 3])/255;

    plot([pointO(1) pointA(1)], [pointO(2) pointA(2)], 'g-', 'LineWidth', 4);
    hold on;
    plot([pointA(1) pointB(1)], [pointA(2) pointB(2)], 'color', OrangeColor, 'LineWidth', 4);
    hold on;
    plot([pointB(1) pointC(1)], [pointB(2) pointC(2)], 'm-', 'LineWidth', 4);
    hold on;
```

```
    plot([pointC(1) pointS(1)], [pointC(2) pointS(2)], 'r-', 'LineWidth', 4);
    hold on;
    plot([pointR(1) pointF(1)], [pointR(2) pointF(2)], 'color', PurpleColor, 'LineWidth', 4);
    hold on;
    plot([pointO(1) pointC(1)], [pointO(2) pointC(2)], 'k-', 'LineWidth', 4);
    hold on;
```

Plot all the points

```
    plot(pointO(1), pointO(2), 'b-o', 'LineWidth', 5);
    text(-50, 30, '  O', 'Color' , 'black' , 'FontSize' ,12)

    plot(pointA(1), pointA(2), 'b-o', 'LineWidth', 5);
    text(pointA(1), pointA(2), '   A', 'Color' , 'black' , 'FontSize' ,12)

    plot(pointB(1), pointB(2), 'b-o', 'LineWidth', 5);
    text(pointB(1), pointB(2), '   B', 'Color' , 'black' , 'FontSize' ,12)

    plot(pointC(1), pointC(2), 'b-o', 'LineWidth', 5);
    text(-65, -350, '    C', 'Color' , 'black' , 'FontSize' ,12)

    plot(pointS(1), pointS(2), 'b-o', 'LineWidth', 5);
    text(pointS(1), pointS(2), '      S', 'Color' , 'black' , 'FontSize' ,12)

    %plot(pointF(1), pointF(2), 'b-o', 'LineWidth', 5);
    text(pointF(1), pointF(2), '   F', 'Color' , 'black' , 'FontSize' ,12)

    plot(pointR(1), pointR(2), 'b-o', 'LineWidth', 5);
    text(pointR(1), pointR(2), '    R', 'Color' , 'black' , 'FontSize' ,12)
```

Trace point R and point F

```
    Grey = '#808080';
    GreyColor = sscanf(Grey(2:end),'%2x%2x%2x',[1 3])/255;

    plot(couplerMidpointTraceXArray, couplerMidpointTraceYArray, 'color', GreyColor, 'LineWidth', 1);

    plot(couplerMidpointTraceXArray1, couplerMidpointTraceYArray1, 'color', GreyColor, 'LineWidth', 1);
```

Draw and plot the slider

```
    DarkGreen = '#006400';
    DarkGreenColor = sscanf(DarkGreen(2:end),'%2x%2x%2x',[1 3])/255;

    angle = (pointF-pointR)/norm(pointF-pointR);
    thetasliding = atan2(angle(2),angle(1));
    s1 = [pointS(1) + 50*cos(thetasliding+(pi/4))   ; pointS(2) + 50*sin(thetasliding+(pi/4))];
    s2 = [pointS(1) + 50*cos(thetasliding+(3*pi/4)) ; pointS(2) + 50*sin(thetasliding+(3*pi/4))];
    s3 = [pointS(1) + 50*cos(thetasliding-(3*pi/4)) ; pointS(2) + 50*sin(thetasliding-(3*pi/4))];
    s4 = [pointS(1) + 50*cos(thetasliding-(pi/4))   ; pointS(2) + 50*sin(thetasliding-(pi/4))];
    plot([s1(1) s2(1) s3(1) s4(1) s1(1)], [s1(2) s2(2) s3(2) s4(2) s1(2)], 'color', DarkGreenColor, 'LineWidth', 3);
    hold on
```

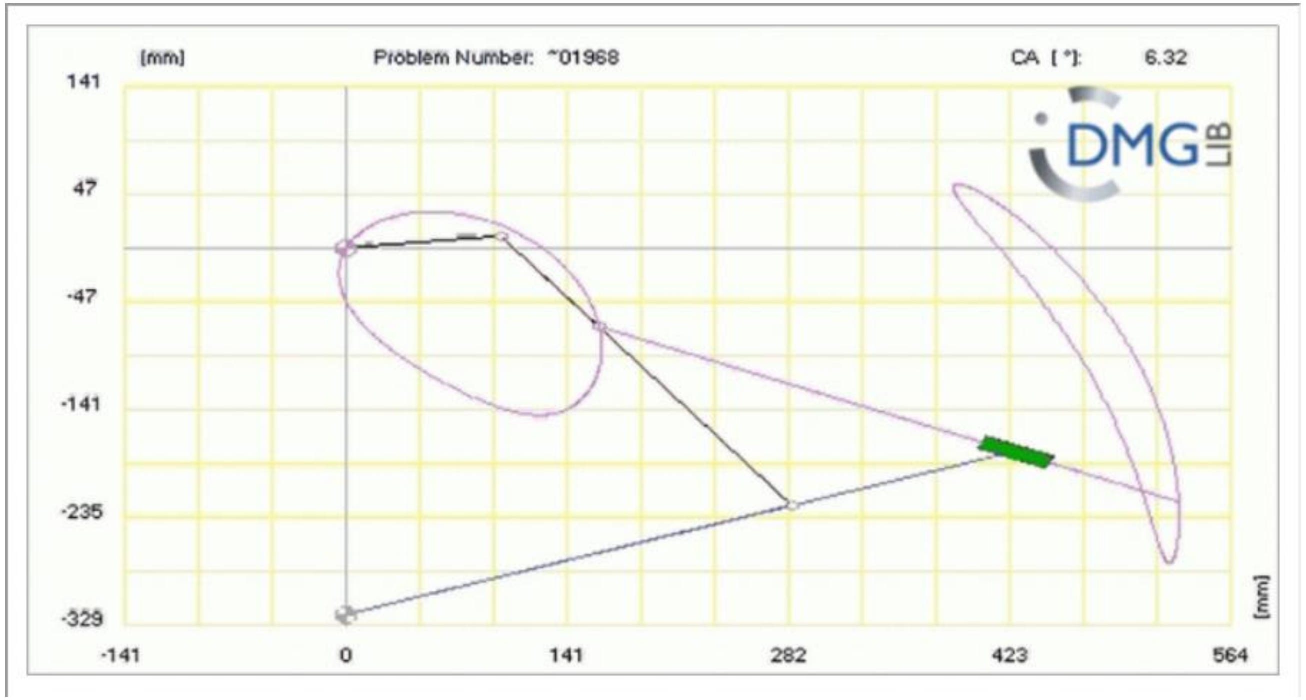
Give labels for the x and y axis and also a title for our animation and figure

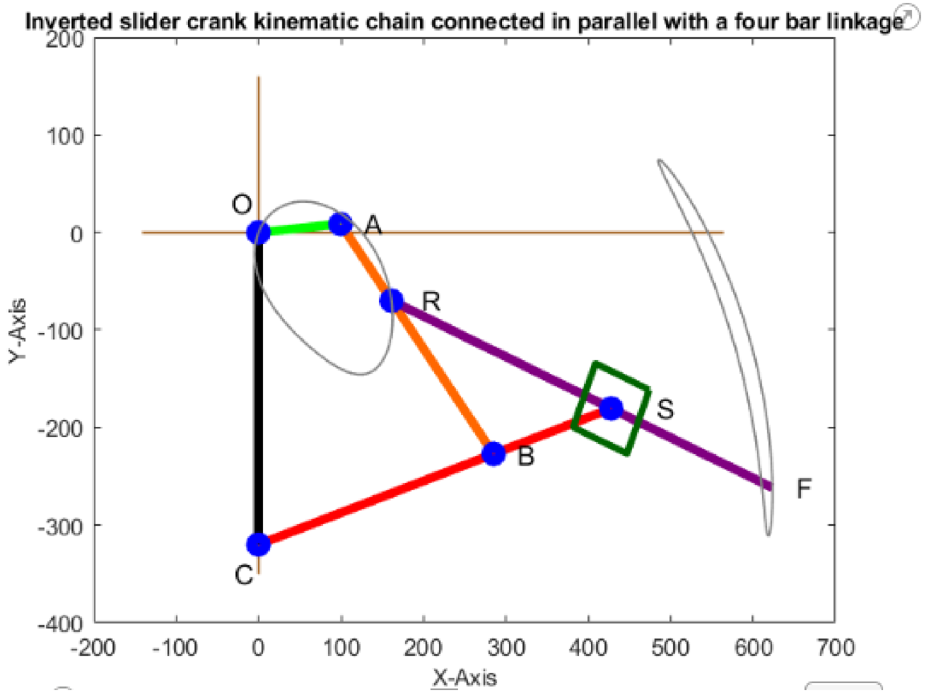
```
    title('Inverted slider crank kinematic chain connected in parallel with a four bar linkage', 'FontSize', 10);
    xlabel('X-Axis', 'FontSize', 10);
    ylabel('Y-Axis', 'FontSize', 10);
```

Use "drawnow()" command to animate

```
    drawnow();
    grid on
    %pause(0.001);
end
```

MATLAB Program: Mechanism Screenshot





MATLAB Program: Mechanism Video