

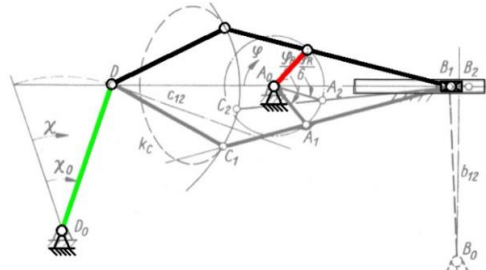
Amrita School of Engineering, Bengaluru

Amrita Vishwa Vidyapeetham

Course: 19PHY113/Computational Engineering Mechanics - 2

Course Instructor: Mr. Rajeevlochana G. Chittawadigi (Dept. of Mech. Engg.)

Course Project: Position, Velocity and Static Force Analysis of Koppelrastgetriebe Mechanism

Team No. and Members	Project Details
No. 10 BL.EN.U4AIE19010 BHUVANASHREE MURUGADOSS BL.EN.U4AIE19027 KARNA SAI NIKHILESH REDDY BL.EN.U4AIE19030 KODE JAI SURYA	 https://www.dmg-lib.org/dmglib/handler?manim=312022

1. Introduction

The course Computational Engineering Mechanics laid the foundation for engineering mechanics by providing a connection between mechanics and computational thinking. Robotics is one of the major areas where AI is applied, hence understanding the fundamentals of Robotics becomes crucial for us, the students of the branch – Artificial Intelligence Engineering.

Hence, mechanisms, which are the fundamental building blocks of robots/machines, need to be studied in detail. Through this project we aim to assess the knowledge we have gained through the course of our year-long study of Computational Engineering Mechanics. We will be applying the knowledge we gained, to perform Position, Velocity and Static Force Analysis of Koppelrastgetriebe Mechanism

2. Description of the Mechanism

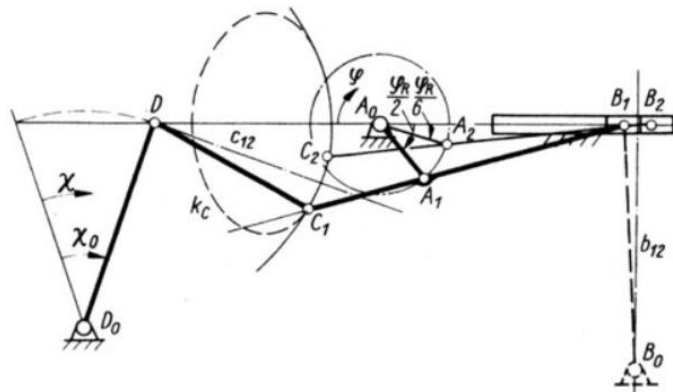


Fig 1: Koppelrastgetriebe Mechanism

Mechanism consists of 6 links and 7 joints.

Links:

- Point A0 and D0 are connected to ground and making them Link #1
- Line joining Point A0 and Point A1 is Link #2 (Crank: Link with 360-degree rotation)
- Line joining Point C1 and Point B1 is Link #3
- Point B1 (slider) is Link #4
- Line joining Point D and C1 is Link #5
- Line joining Point D and D0 is Link #6

Joints:

- 1) Link #1 (point A0) and Link #2 has joint at point A0 (ground link's Hinge). R_{12}
- 2) Link #2 and #3 has joint at point A1. R_{23}
- 3) Link #3 and #4 has joint at point B1. P_{34}
- 4) Link #5 and #3 has joint at point C1. R_{35}
- 5) Link #6 and #5 has joint at point D. R_{56}
- 6) Link #1 (point D0) and Link #2 has joint at point D0. R_{51}
- 7) Link #3 and Link #4 has joint at point B1. R_{34}

Grubler's Equation for a Planar Mechanism:

$$DOF = 3(N - 1) - 2P_1 - P_2$$

Where:

DOF = Number of degrees of freedom

N = total number of links in the mechanism

P_1 = total number of lower pairs.

P_2 = total number of higher pairs.

For Koppelrastgetriebe Mechanism:

$N = 6$ (4 Links + 1 Ground Link + 1 Slider)

$P_1 = 7$ (6 Revolute joints + 1 Prismatic Joint)

$P_2 = 0$ (No Higher Pairs)

$$DOF = 3(6 - 1) - 2 * 7 - 0$$

Degree of Freedom for above Mechanism using Grubler's Equation is 1.

$$DOF = 1$$

Physical Significance of 1 DOF with respect to this mechanism: -

- The number of independent variables required to define the system uniquely is one. When the independent input parameter is changed all other links follow or they are dependent on the input link.
- Entire mechanism is controlled with one input parameter. Rotatory motion can be given as input for this mechanism at Link #2 which satisfies the DOF.

Steps taken to trace the Mechanism's image:

- 1) We pasted the screenshot of the mechanism from the website.
- 2) We aligned the image such that point A0, coincided with origin and by making sure that aspect ratio remained the same.
- 3) We traced the image canvas by drawing lines over the links.
- 4) We measured the angle that link A0A1 makes with the horizontal, length of each link and found co-ordinates for point D0 and A0.

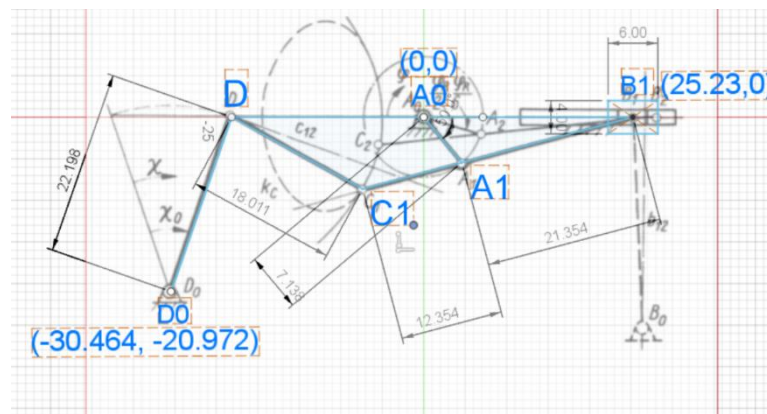


Fig 2: Trace of Position Analysis from the Mechanism's image

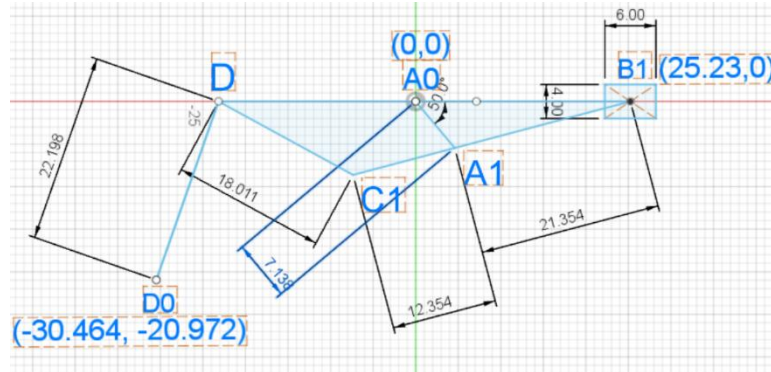


Fig 3: Position Analysis of Koppelrastgetriebe Mechanism without Trace

We are attempting to perform position, velocity and static force analysis for Koppelrastgetriebe Mechanism using geometric approach.

3. Computational Geometry

Parametric Equation of a line: -

Any point P on a line can be written as:

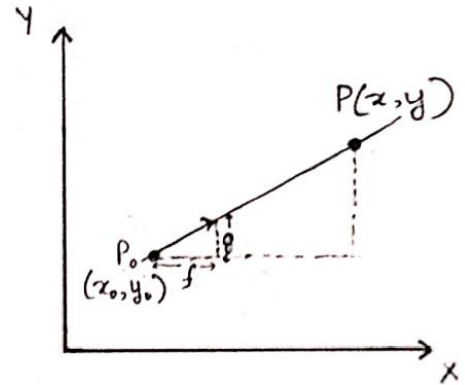
$$\vec{P} = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} + t * \begin{bmatrix} f \\ g \end{bmatrix} \dots (1)$$

Where 't' is a scalar and is called the 'parameter' that changes for different points along the line.

$\begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$ = co-ordinates of the initial points

t = length of the line

$\begin{bmatrix} f \\ g \end{bmatrix}$ = components of the direction vector; slope of the line = g/h.



Intersection of two lines (represented in parametric form): -

Equation of Line 1:

$$\vec{p} = \vec{p}_A + t_A(\vec{w}_A) \dots (2)$$

Equation of Line 2:

$$\vec{p} = \vec{p}_B + t_B(\vec{w}_B) \dots (3)$$

The above two equations are equated as they intersect at a point:

$$\vec{p}_A + t_A(\vec{w}_A) = \vec{p}_B + t_B(\vec{w}_B) \dots (4)$$

$$\vec{p} = \vec{p}_A + t_A(\vec{w}_A) = \vec{p}_B + t_B(\vec{w}_B) \dots (5)$$

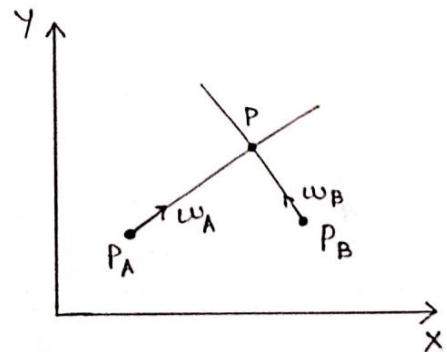
We have two unknowns: t_A and t_B which we solve for by the following two equation using Cramer's Rule:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x_A \\ y_A \end{bmatrix} + t_A \begin{bmatrix} f_A \\ g_A \end{bmatrix} = \begin{bmatrix} x_B \\ y_B \end{bmatrix} + t_B \begin{bmatrix} f_B \\ g_B \end{bmatrix} \dots (6)$$

$$\begin{bmatrix} f_A - f_B \\ g_A - g_B \end{bmatrix} \begin{bmatrix} t_A \\ t_B \end{bmatrix} = \begin{bmatrix} x_A - x_B \\ y_A - y_B \end{bmatrix} \dots (7)$$

$$Ax = b \dots (8)$$

$$\det = \begin{vmatrix} f_A & -f_B \\ -g_B & -g_A \end{vmatrix} = f_B g_A - f_A g_B \dots (9)$$



By Cramer's Rule:

$$t_A = \frac{\begin{vmatrix} x_B - x_A & -f_B \\ y_B - y_A & -g_B \end{vmatrix}}{\det} \dots (10)$$

$$t_A = \frac{f_B(y_B - y_A) - g_B(x_B - x_A)}{\det} \dots (11)$$

$$t_B = \frac{\begin{vmatrix} f_A & x_B - x_A \\ g_A & y_B - y_A \end{vmatrix}}{\det} \dots (12)$$

$$t_B = \frac{f_A(y_B - y_A) - g_A(x_B - x_A)}{\det} \dots (13)$$

Once t_A or t_B is found, the point of intersection can be found by substituting its value in any of the line equations:

Equation of Line 1:

$$\vec{p} = \vec{p}_A + t_A(\vec{w}_A) \dots (14)$$

Equation of Line 2:

$$\vec{p} = \vec{p}_B + t_B(\vec{w}_B) \dots (15)$$

Line – Circle Intersection: -

$$\text{Circle Equation: } (x - x_c)^2 + (y - y_c)^2 = r^2 \dots (16)$$

Parametric Circle Equation:

$$x = x_c + r \cos \theta \dots (17)$$

$$y = y_c + r \sin \theta \dots (18)$$

$$\vec{p}_c = \begin{bmatrix} x_c \\ y_c \end{bmatrix} \dots (19)$$

Considering \vec{p} as intersection point (there might be two or less)

Point \vec{p} with respect to the line's equation: -

$$\vec{p} = \vec{p}_L + t(\vec{w}_L) \dots (20)$$

Point \vec{p} with respect to the circle's equation: -

$$\vec{p} = \vec{p}_L + \begin{bmatrix} r \cos \theta \\ r \sin \theta \end{bmatrix} \dots (21)$$

The above two equations are equated as they intersect at a point (eq 20 & 21)

$$\vec{p} + t\vec{w}_L = \vec{p}_c + \begin{bmatrix} r \cos \theta \\ r \sin \theta \end{bmatrix} \dots (22)$$

$$\begin{bmatrix} x_L \\ y_L \end{bmatrix} + t \begin{bmatrix} f_L \\ g_L \end{bmatrix} = \begin{bmatrix} x_c \\ y_c \end{bmatrix} + \begin{bmatrix} r \cos \theta \\ r \sin \theta \end{bmatrix} \dots (23)$$

We have 2 parameter t & θ , so we try to get rid of one

$$(x_L - x_c) + t f_L = r \cos \theta \dots (24)$$

$$(y_L - y_c) + t g_L = r \sin \theta \dots (25)$$

Square and add (1) & (2)

$$(x_L - x_c)^2 + t^2 f_L^2 + 2(x_L - x_c)(t f_L) + (y_L - y_c)^2 + t^2 g_L^2 + 2(y_L - y_c)(t g_L) = r^2 \dots (26)$$

Only 1 parameter – t is remaining

2 roots of the above equation give (eq 26):

$t \rightarrow t_A \Rightarrow$ Exists corresponding θ_A

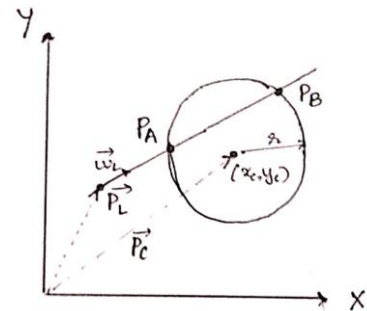
$t \rightarrow t_B \Rightarrow$ Exists corresponding θ_B

$$t = \frac{f_L(x_c - x_L) - g_L(y_c - y_L)}{t_L^2 + g_L^2} \pm \sqrt{x^2(t_L^2 + g_L^2) - [f_L(y_c - y_L) - g_L(x_c - x_L)]} \dots (27)$$

$$t = t_{exp1} \pm t_{exp2} \dots (28)$$

The 2 solutions of t are:

$$t_A = t_{exp1} + t_{exp2} \dots (29)$$



$$t_B = t_{exp1} - t_{exp2} \dots (30)$$

Note:

When the discriminant of t_{exp2} is $< 0 \rightarrow$ there exists no intersection point

When the discriminant of t_{exp2} is $= 0 \rightarrow$ there exists only one intersection point

When the discriminant of t_{exp2} is $> 0 \rightarrow$ there exists two intersection points

If there exists two/one points of intersection we get them by substituting t_A and t_B value in the following equations:

$$\vec{p}_A = \vec{p}_L + t_A(\vec{w}_L) \dots (31)$$

$$\vec{p}_B = \vec{p}_L + t_B(\vec{w}_L) \dots (32)$$

θ_A & θ_B need not be found

Circle - Circle Intersection:

We assume that a line passes through the two intersection points of the circle, a common chord, once we find the equation of that line, the problem can be converted to a line-circle intersection algorithm

Triangle $P_A C_A P$ and Triangle $P_A C_B P$ have $P_A P$ in common

Let:

$$C_A C_B = l_C \dots (33)$$

$$C_A P = m \dots (34)$$

By applying Pythagoras Formula:

$$P_A P^2 = C_A P_A^2 - C_A P^2 \dots (35)$$

$$P_A P^2 = r_A^2 - m^2 - 1 \dots (36)$$

$$P_A P^2 = C_B P_A^2 - C_B P^2 \dots (37)$$

$$P_A P^2 = r_B^2 - (l_C - m)^2 - 2 \dots (38)$$

Equation 1 and 2

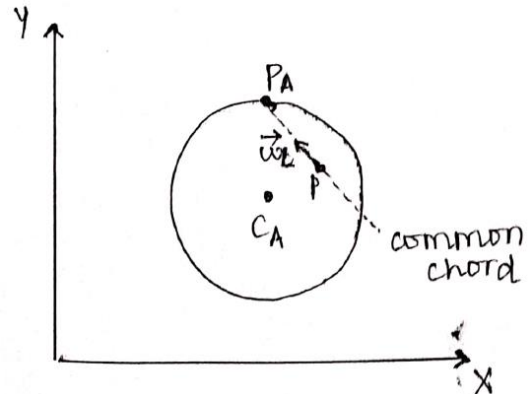
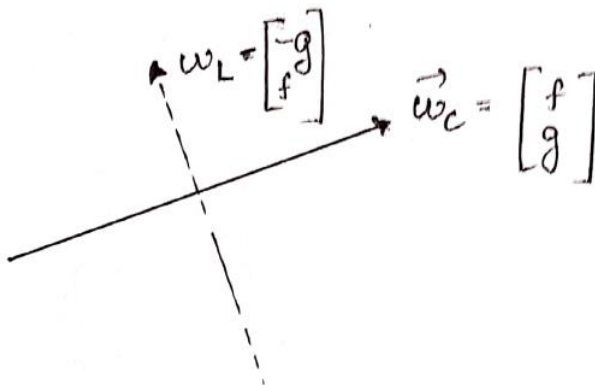
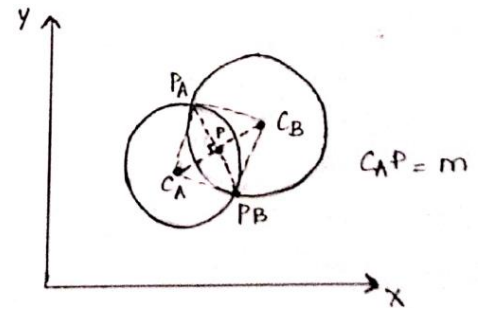
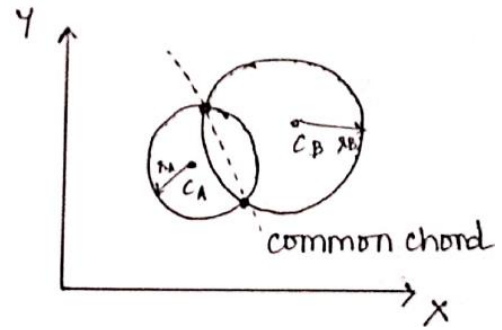
$$r_B^2 - r_C^2 - l_B^2 + 2l_C m + m^2 - m^2 = 0 \dots (39)$$

The above equation gives us:

$$m = \frac{r_A^2 - r_B^2 + l_C^2}{2l_C} \dots (40)$$

$$\vec{p} = \vec{C}_A + \frac{(\vec{C}_B - \vec{C}_A)}{|\vec{C}_B - \vec{C}_A|} \cdot m \dots (41)$$

$$\vec{w}_C = \frac{(\vec{C}_B - \vec{C}_A)}{|\vec{C}_B - \vec{C}_A|} - \left[\frac{f}{g} \right] \dots (42)$$



Since P and \vec{w}_L is known, we can compute the intersection points of the two circles - P_A and P_B using the Line-Circle Intersection method (mentioned in the previous section).

4. Position Analysis

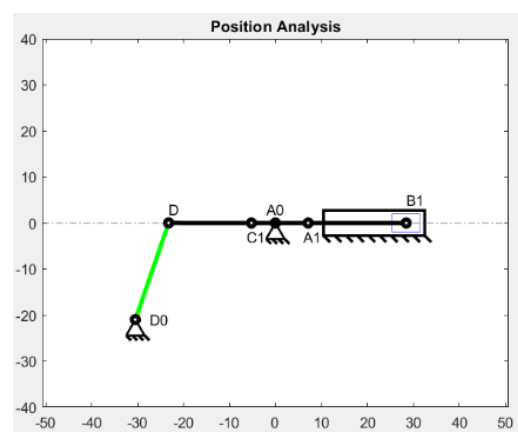
Line-Circle Intersection: To get point of the slider, we provided Point A1 coordinates, Sliding path and Length of the Follower.

Circle-Circle Intersection: To get Point D, we provided Point C1, length of Link #5(C1D), Point D0 and length of Link #6(D0D).

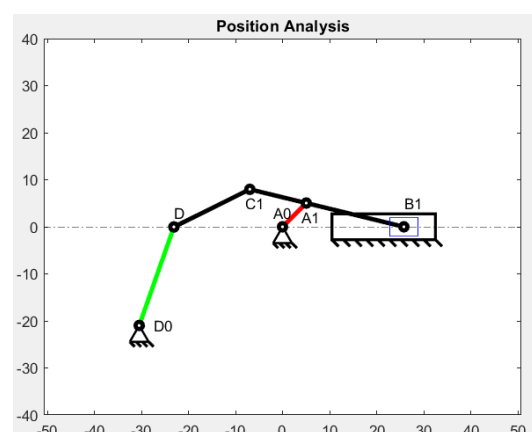
Steps taken in writing the Position Analysis MATLAB code:

- 1) Link description and Input angle of Crank:
Initialize all the variables like Link lengths, Crank angle and coordinates of Point A0 and Point D0.
Covert input angle from degrees to radians to maintain consistency.
- 2) Determining the Point A1:
Position of A1 with respect to A0 was obtained by using below formula.
$$PointA1 = PointA0 + (Length\ of\ A0A1) * ([\cos(AnlgeA0)\ \sin(angleA0)])$$
- 3) Determining the Point B1:
Point B1 was determined by drawing construction line and a circle and finding their point of intersection using Line-Circle Intersection Algorithm.
- 4) Determining the Point C1:
Point C1 was determined by extending a line along a direction opposite to the vector A1B1.
$$Length\ of\ A1B1 = norm(pointA1 - pointB1)$$
$$Direction\ of\ A1B1 = (PointA1 - PointB1)/(length\ of\ A1B1)$$
$$PointC1 = PointA1 + (Length\ of\ C1A1) * (Direction\ of\ A1B1)$$
- 5) Determining the Point D:
Point D was determined by drawing two construction circle's and finding their point of intersection using Line-Circle Intersection Algorithm
- 6) Plotting all links:
Plot links using plot() function by adding appropriate properties to it. (ex: LineWidth, MarkerEdgeColor, MarkerFaceColor, MarkerSize)
- 7) Plot Traces:
Plot Traces of Points A1, B1, C1 and D by storing their old position into an array.
- 8) Plot Ground Links:
At Point A0 and D0, add Ground Links by using GroundLink() function by passing few parameters. (ex: coordinates, radius, side_len and etc)

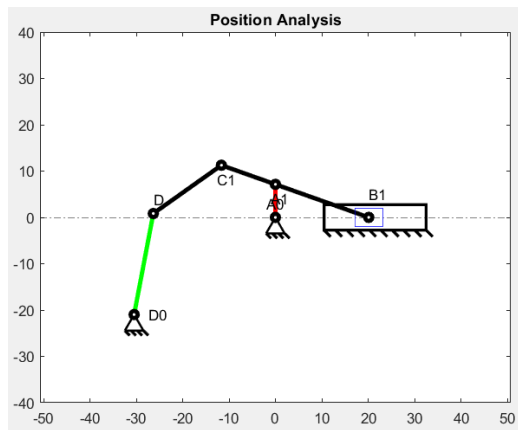
Some screenshots for different values of input joint motion:



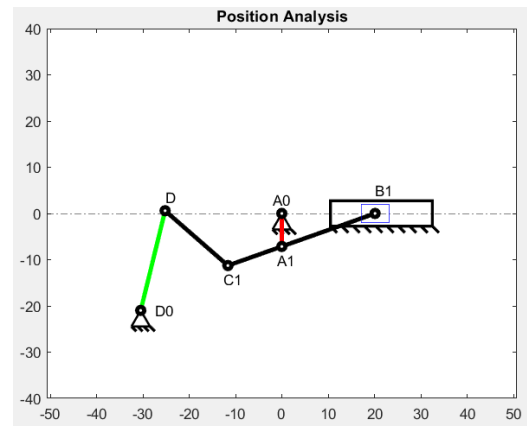
Crank Angle 0 degrees



Crank Angle 45 degrees



Crank Angle 90 degrees



Crank Angle 270 degrees

For input Crank angle of 310 degrees:

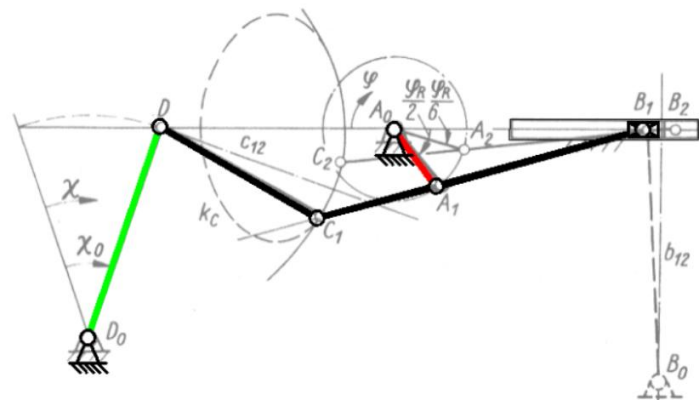


Fig: Koppelrastgetriebe Mechanism from website

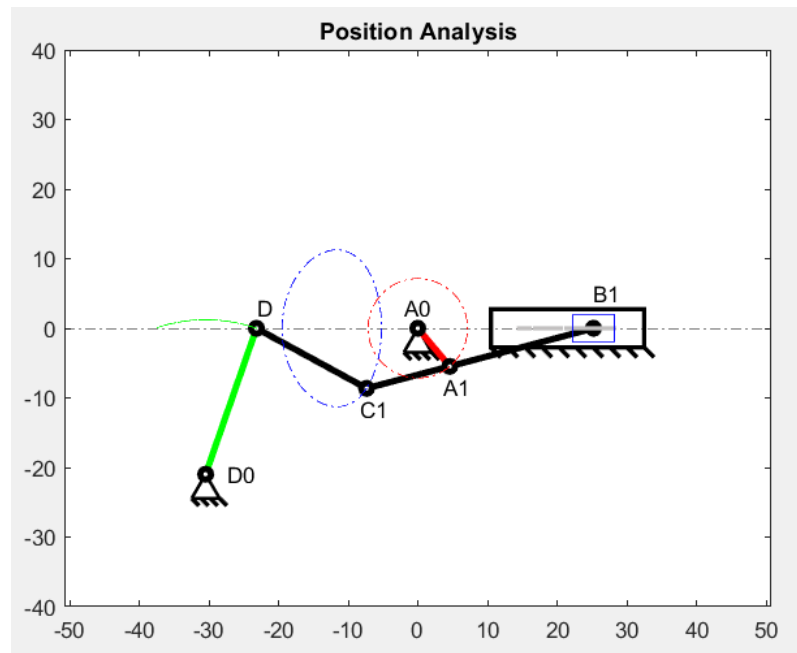


Fig: Koppelrastgetriebe Mechanism using MATLAB

5. Velocity Analysis

Line-Line Intersection:

- 1) To get point b1 in velocity polygon, we provided a construction line perpendicular to link A1B1 and passing through A1; a construction line parallel to x-axis and passing through A0.
- 2) To get point d in velocity polygon, we provided a construction line perpendicular to link C1D and passing through C1; a construction line perpendicular to link D0D and passing through D0.

Formulation of the steps taken to perform Velocity Analysis:

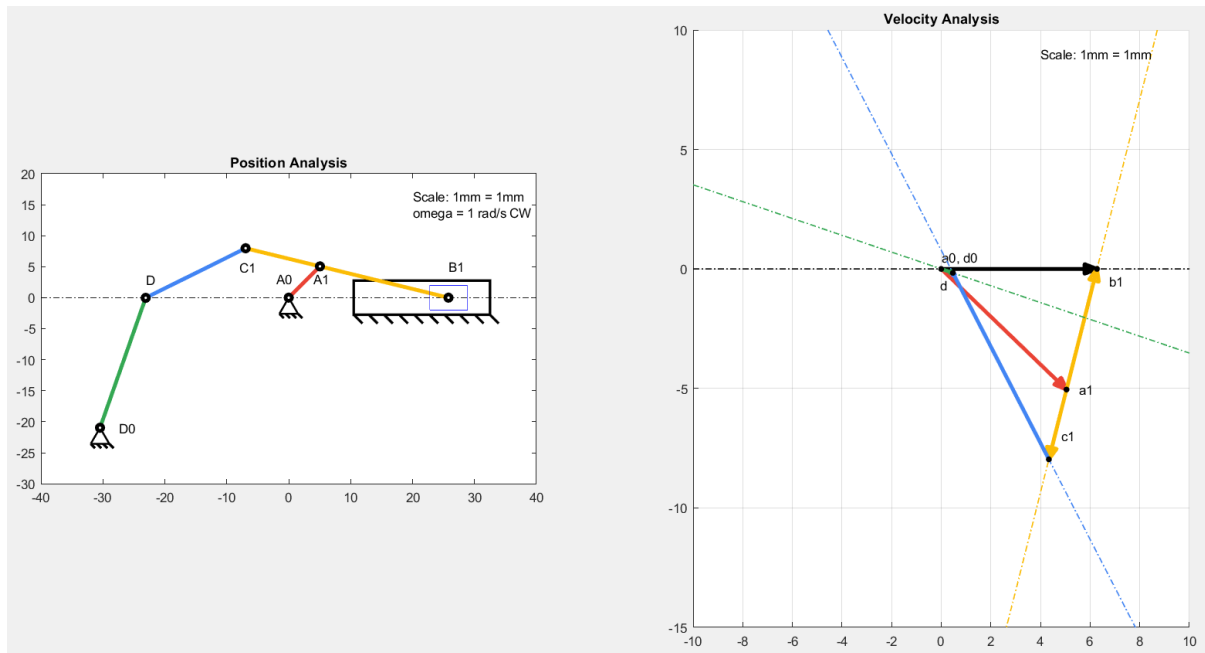


Fig: Velocity Analysis Output with 45 degree Crank input angle.

Steps taken in writing the Velocity Analysis MATLAB code:

- 1) Utilizing the results from the position analysis:
The results obtained from the position analysis – location and orientation/inclination of various links for a given input link angle were stored in appropriate variables.
- 2) Plotting the fixed links:
Points a0 and d0 were fixed at the origin as they are the grounded links
- 3) Determining the Point a1:
Velocity of a1 with respect to a0 was obtained by using the formula:

$$V = r * \omega \rightarrow V_{a1/a0} = \text{length}(A0A1) * (1); \text{ here } \omega = 1 \text{ rad/s}$$
- 4) Determining the Point b1:
Point b1 was determined by drawing two construction lines and finding their point of intersection using the Line-Line Intersection Algorithm:
 - Point b1 with respect to a0 has a velocity that is parallel to the X-axis which is its direction of sliding, hence a horizontal construction line passing through point a0 was plotted.
 - Point b1 with respect to point a1 has a velocity that's perpendicular to the input link A0A1(link #2), hence a construction line perpendicular to A0A1 was plotted.
 - The intersection of the above constructed lines gives the location of point b1 and the magnitude of the vector from a0 to b1 and vector from a1 to b1 was determined.

$$\text{magnitude of vector } [x_component; y_component] = \sqrt{(x_component)^2 + (y_component)^2}$$

5) Determining the point c1:

Point c1 will be located along the line joining a1 and b1 since all three of them are on the same link, however the direction of c1 with respect to a1 will be opposite to the direction of b1 with respect to a1. Magnitude of the vector a1c1 was determined by equating the angular velocity of link C1B1 as given below

$$\omega = \frac{V}{r}$$

$$\frac{V_{c1/a1}}{A1C1} = \frac{V_{b1/a1}}{A1B1}$$

$$V_{c1/a1} = \frac{V_{b1/a1}}{A1B1} * A1C1$$

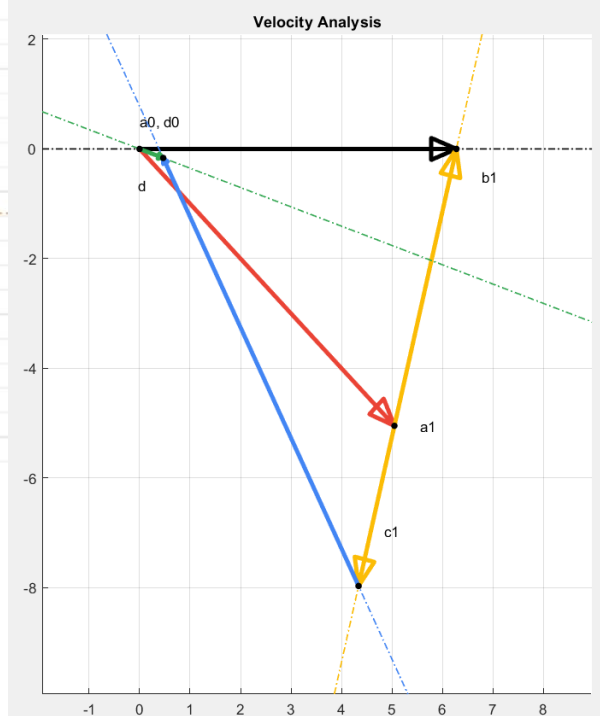
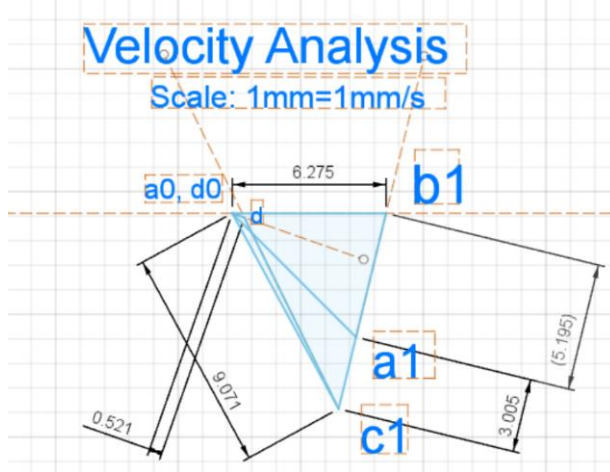
6) Determining the point d:

Point d was determined by drawing two construction lines and finding their point of intersection using the Line-Line Intersection Algorithm:

- Point d with respect to point c1 has a velocity that's perpendicular to the link DC1(link #5), hence a construction line perpendicular to DC1 was plotted.
- Point d with respect to fixed point d0 has a velocity that's perpendicular to the link DD0(link #2), hence a construction line perpendicular to DD0 was plotted.
- The intersection of the above constructed lines gives the location of point d and the magnitude of the vector from a0 to d and vector from c1 to d was determined using the norm function or the formula

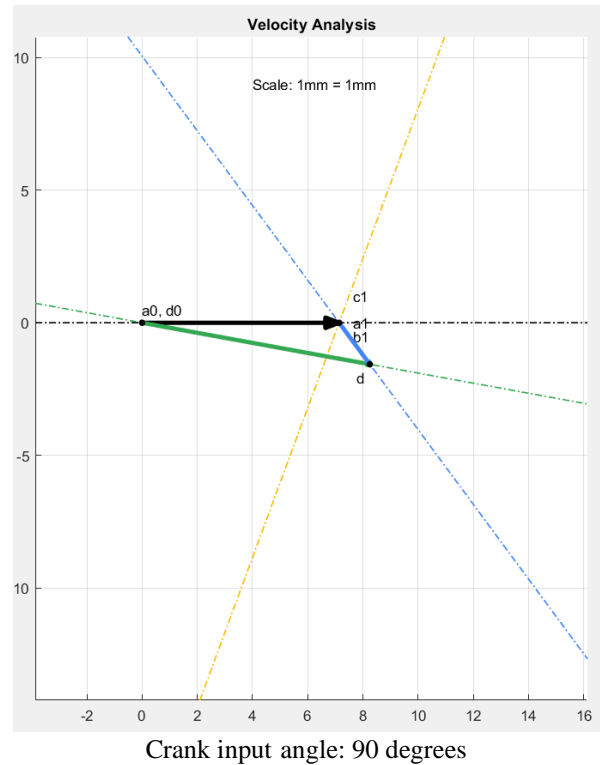
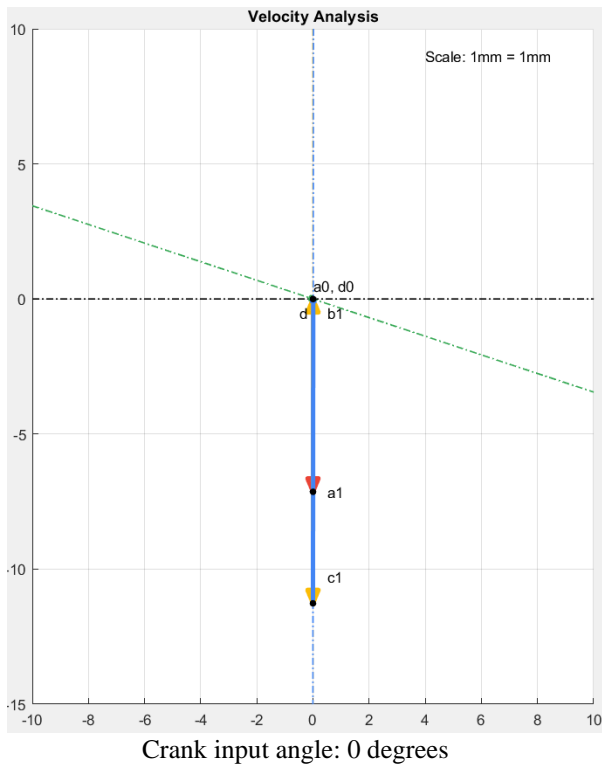
$$\text{magnitude of vector } [x_component; y_component] = \sqrt{(x_component)^2 + (y_component)^2}$$

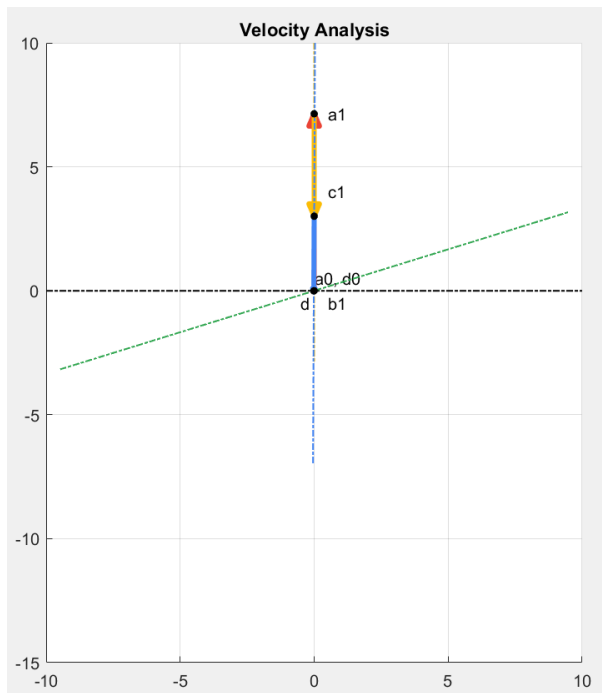
Validation of our Mechanism's Velocity Analysis MATLAB code with Fusion (input angle: 45 degrees)



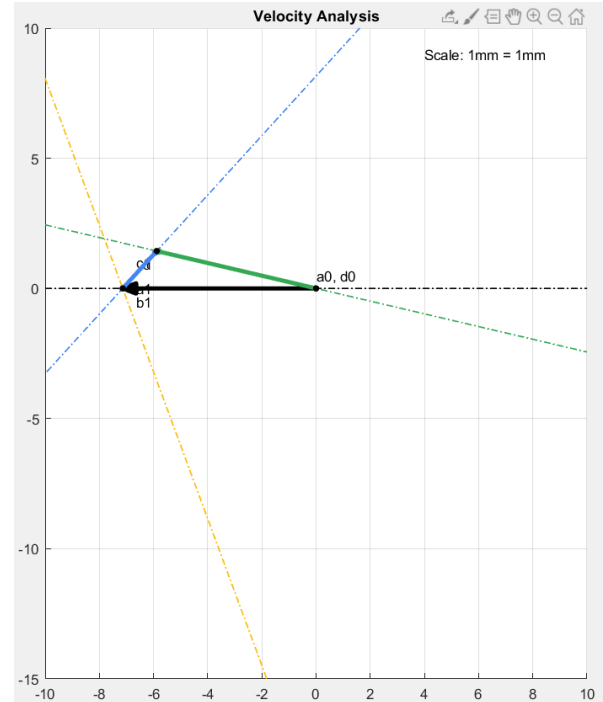
Result Obtained from MATLAB:	Result Obtained from Fusion Model:
<p>Absolute Velocities:</p> <p>Velocity of a1 wrt a0 = 7.1380 mm/s</p> <p>Velocity of b1 wrt a0 = 6.2751 mm/s</p> <p>Velocity of c1 wrt a0 = 9.0713 mm/s</p> <p>Velocity of d wrt a0 = 0.4964 mm/s</p> <p>Relative Velocities:</p> <p>Velocity of b1 wrt a1 = 5.1945 mm/s</p> <p>Velocity of c1 wrt a1 = 3.0052 mm/s</p> <p>Velocity of d wrt c1 = 8.7090 mm/s</p>	<p>Absolute Velocities:</p> <p>Velocity of a1 wrt a0 = 7.138 mm/s</p> <p>Velocity of b1 wrt a0 = 6.275 mm/s</p> <p>Velocity of c1 wrt a0 = 9.071 mm/s</p> <p>Velocity of d wrt a0 = 0.521 mm/s</p> <p>Relative Velocities:</p> <p>Velocity of b1 wrt a1 = 5.195 mm/s</p> <p>Velocity of c1 wrt a1 = 3.005 mm/s</p> <p>Velocity of d wrt c1 = 8.692 mm/s</p>

Results for a few input joint angles:





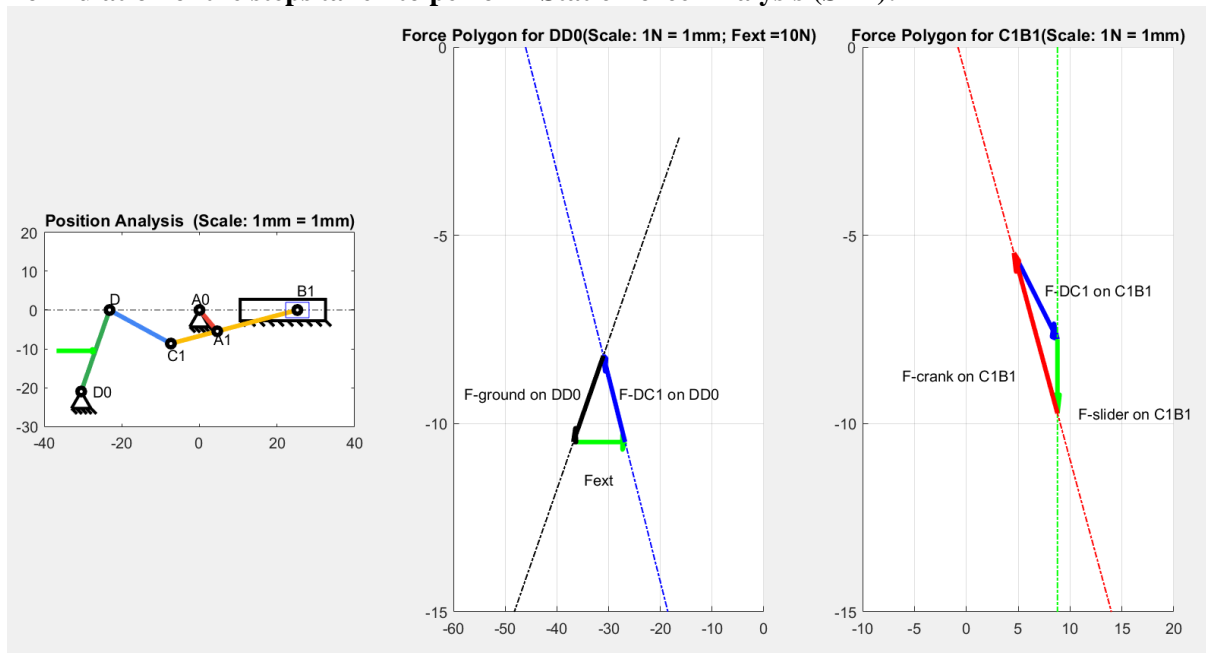
Crank input angle: 180 degrees



Crank input angle: 270 degrees

6. Static Force Analysis

Formulation of the steps taken to perform Static Force Analysis (SFA):



Steps taken in writing the Velocity Analysis MATLAB code:

- 1) Utilizing the results from the position analysis
The results obtained from the position analysis – location and orientation/inclination of various links for a given input link angle were stored in appropriate variables. And for showing the applied external force can be observed.
- 2) Free Body Diagrams of each link to determine the relationship between the applied force and the reaction forces at each of the joints.

Fig. FBD of DD0 and DC1

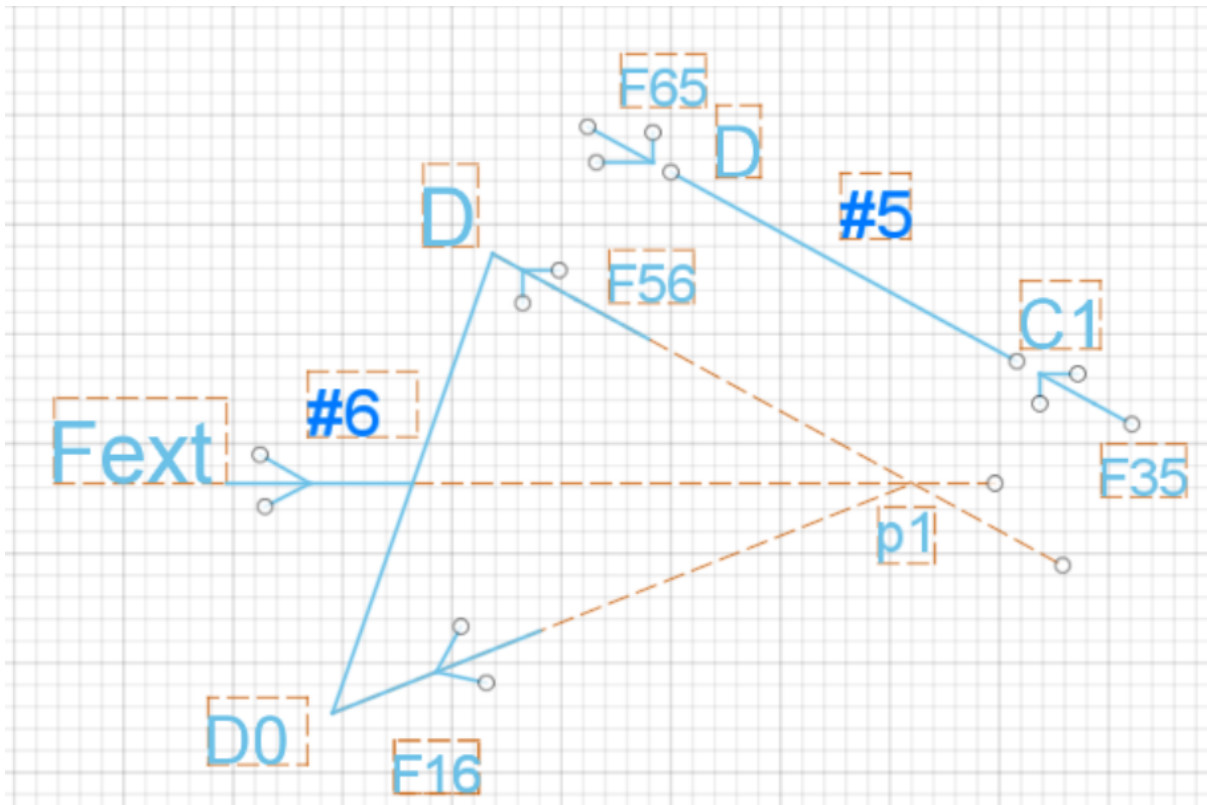


Fig. Force Polygon For forces on DD0

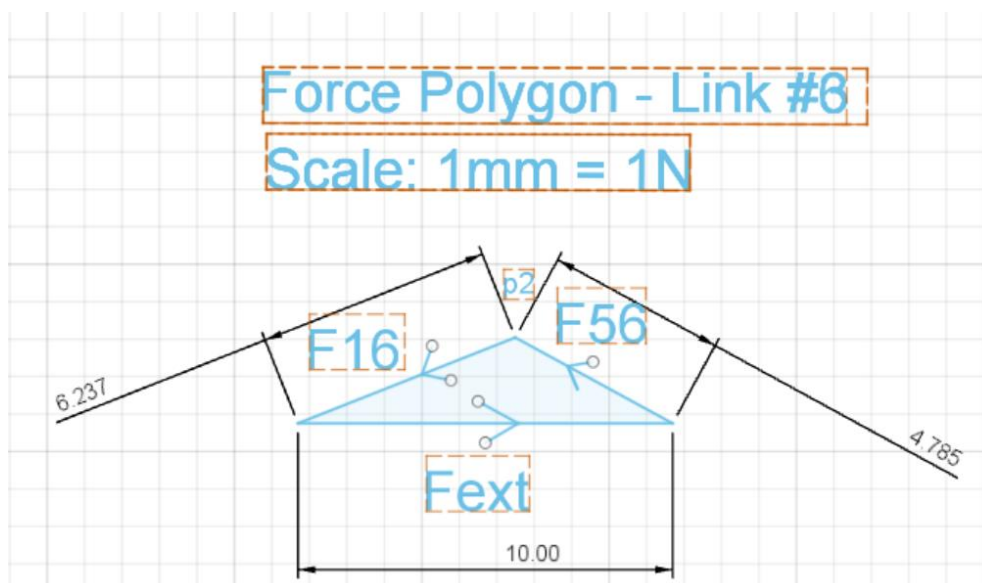


Fig. FBD for C1B1

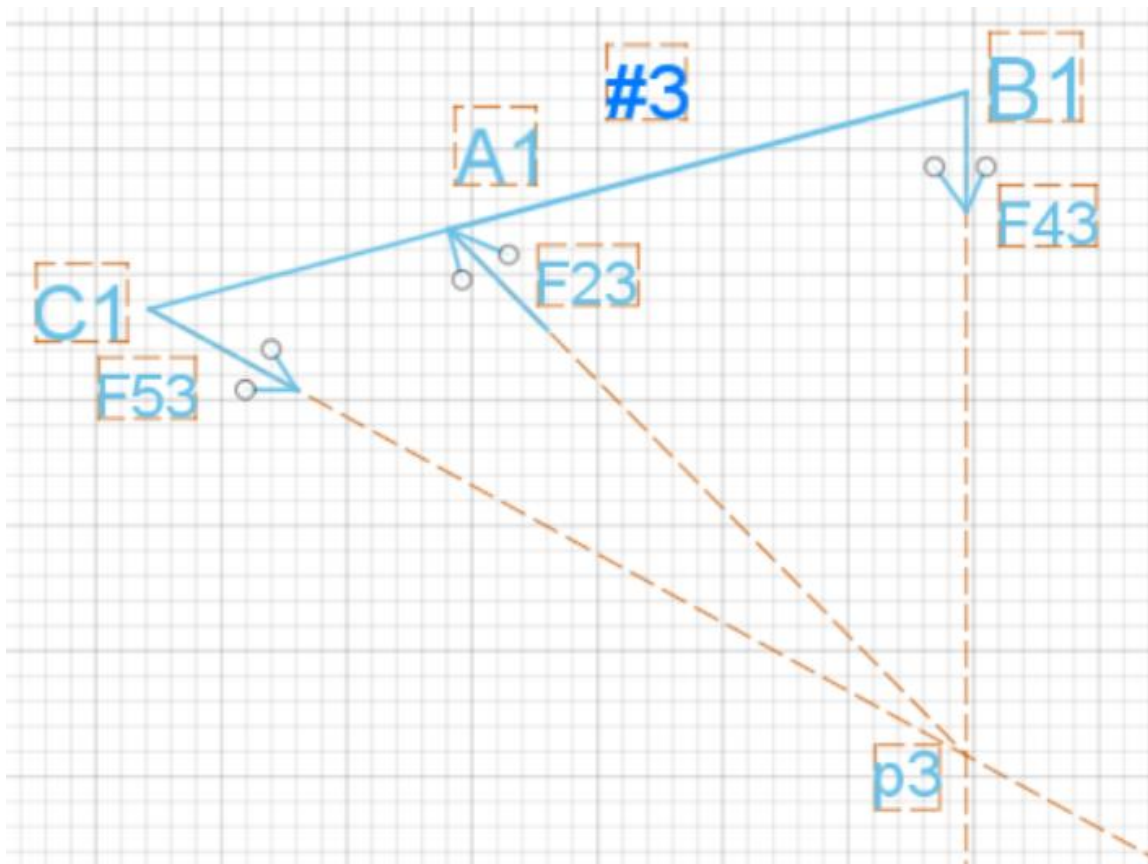
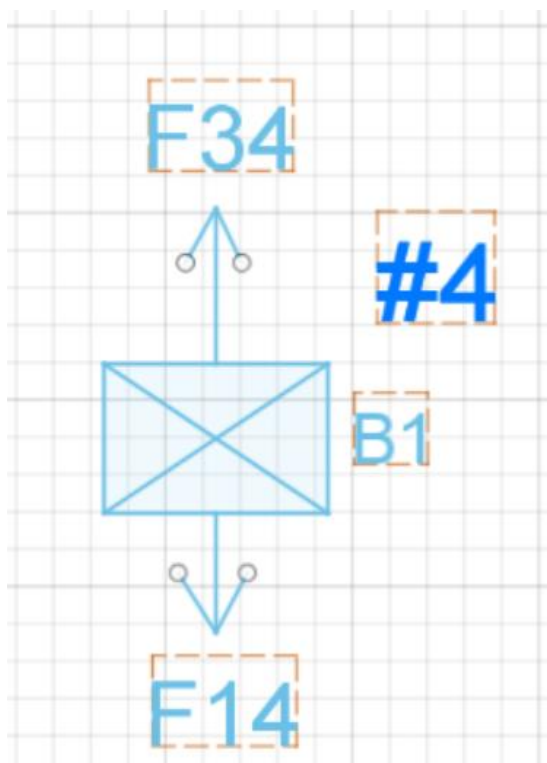
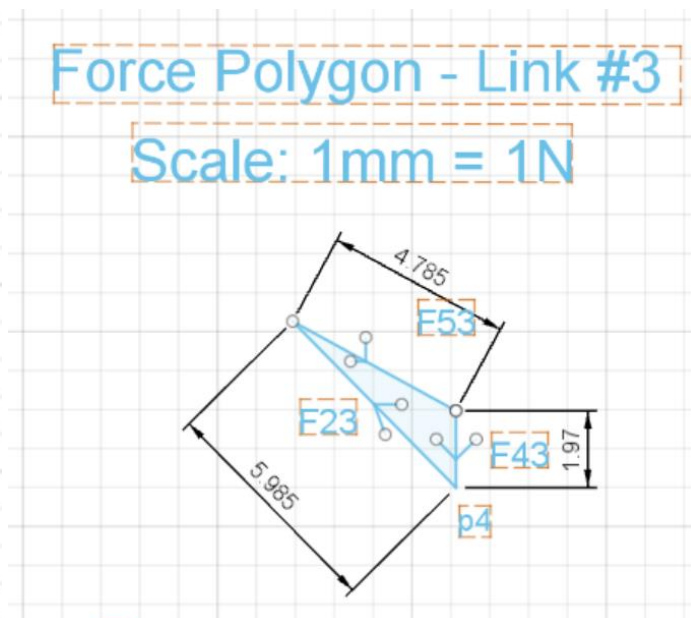


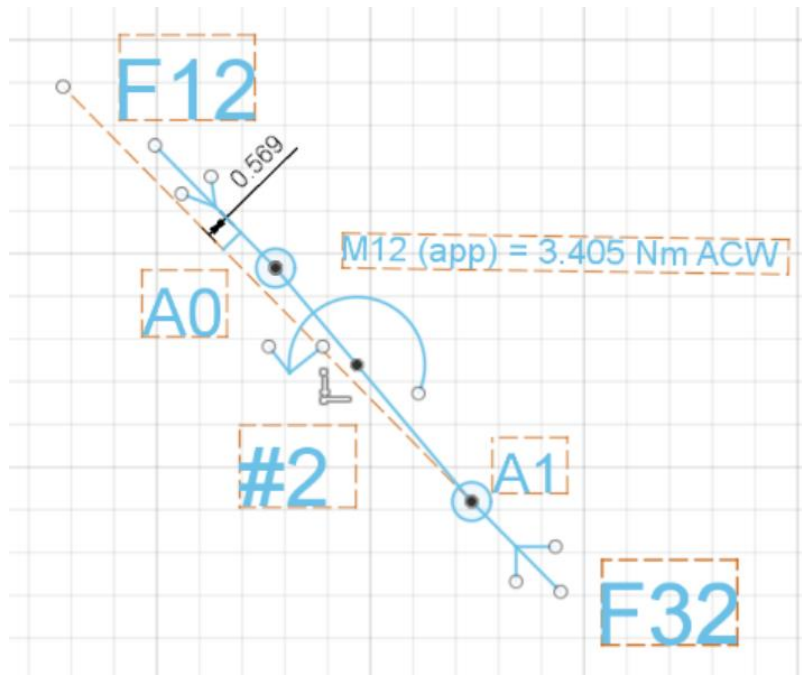
Fig.FBD of Slider:



Force Polygon for Forces acting on C1B1



FBD for Crank – Link #2, where moment is applied



- 3) The MATLAB coding part.
The magnitude of the force, the angle at which it acts and the point where it acts have all been initialized. The directions of some of the links have been stored from the position analysis.
- 4) The directions of the forces on the link with applied force are determined.

The direction of the unknown forces is found out using Line-Line Intersection function Algorithm. The force polygon is determined, the unknown reaction forces on that link are determined by the relationship between the links and joints. For point p1 we have used the Line-Line Intersection function which gave us p1 from which we were able to find the direction of reaction force of ground on D0D which should pass through the point. Similarly, we found p3 by Line-Line Intersection function from which we were able to find the direction of reaction force which passes through p3 to make the link stay in static equilibrium and gave us the direction of the force.

- 5) Determining the magnitude of unknown forces.

The magnitude of unknown forces is determined by the force polygons plotted in MATLAB. For which we have considered the directions which we have got from the Free body diagrams and made the force polygons, from which the point of intersection was found out by using Line-Line Intersection function. For the force polygon for the forces acting on DD0, we drew the force external and then were able to draw the construction lines in the direction of reaction force of DC1 on DD0 and then we also drew the construction lines for the reaction force of ground on DD0 and we have seen that they have intersected at a single point which we were able to find out by using Line-Line Intersection function we found p2 and then found out the magnitude of each force using norm function. After we found the reaction force of C1D on DD0 we used the magnitude on link C1B1 as link C1D is a 2-force member. Now using the directions of reaction forces acting on C1B1 we drew the construction lines for the reaction forces. We can see that they intersect at a point and using Line-Line Intersection function found out p4 from which we got the magnitudes of each reaction force.

- 6) Determining the Moment.

After finding out all of the forces. We also find out that there is a resultant moment on the input crank A0A1. Which we need to nullify by adding a moment in the opposite direction of the resultant moment.

We know that moment acting on input crank A0A1 is equal to the product of magnitude of the force and perpendicular distance between them. We use that to find the resultant force and then add a opposing moment to it.

Note that the force acting on A0A1 is the reaction force which is acting on the Link C1B1 at the point A1. We also know the direction of this force. We find the angle between r = the force and the link by using

$$\begin{aligned}\text{CosTheta} &= (\text{dot}(u,v)/(\text{norm}(u)*\text{norm}(v))); \\ \text{ThetaInDegrees} &= \text{real}(\text{acosd}(\text{CosTheta}));\end{aligned}$$

$$\text{Moment} = \text{Force} * \text{Perpendicular Distance (or)} \text{ Force} * \text{Link Length} * \text{Sin(Angle between force and the link)}$$

Validation of our Mechanism's Static Force Analysis MATLAB code with Fusion results (Fext = 10N, along +ve X-axis)

Result Obtained from MATLAB:	Result Obtained from Fusion Model:
<p>The applied external force on DD0 is: 10.00 N The reaction force of C1D on DD0 is: 4.7865 N The reaction force of ground on DD0 is: 6.2343 N The reaction force of C1D on C1B1 is: 4.7865 N The reaction force of slider on C1B1 is: 1.9699 N The reaction force of A0A1 on C1B1 is: 5.9850 N The reaction force of C1B1 on A0A1 is: 5.9850 N The moment acting on A0A1 is: -3.424596 Nm in Clockwise sense. We need to apply a moment of 3.424596 Nm in Anti-Clockwise sense on A0A1 to keep the mechanism in Static Equilibrium</p>	<p>The applied external force on DD0 is: 10.00 N The reaction force of C1D on DD0 is: 4.785 N The reaction force of ground on DD0 is: 6.237 N The reaction force of C1D on C1B1 is: 4.785 N The reaction force of slider on C1B1 is: 1.97 N The reaction force of A0A1 on C1B1 is: 5.985 N The reaction force of C1B1 on A0A1 is: 5.985 N The moment acting on A0A1 is: -3.42 Nm in Clockwise sense. We need to apply a moment of 3.405 Nm in Anti-Clockwise sense on A0A1 to keep the mechanism in Static Equilibrium</p>

7.Conclusion:

We have performed Position, Velocity and Static Force Analysis of Koppelrastgetriebe Mechanism in MATLAB and verified the results using geometric drawings in Fusion360 Through the course of developing the project we have gained a basic understanding on how a mechanism can be analysed to know its working.

Also, we have developed skills to design and animate a given mechanism's position, velocity and static force analysis.