ME2201 T8

Anton Beny M S, ME23B015

October 2024

1 Radial Roller: Graphical Method

For the given follower motion, the equations for rise and return is given by:

$$y_{rise}(\theta) = 40\left(\frac{\theta}{180^\circ} - \frac{1}{2\pi}\sin\left(2\pi\frac{\theta}{180^\circ}\right)\right)$$

$$y_{return}(\theta) = 40 - 40\left(\frac{\theta - 180^\circ}{180^\circ} - \frac{1}{2\pi}\sin\left(2\pi\frac{\theta - 180^\circ}{180^\circ}\right)\right)$$

1.1 Geogebra

I used Geogebra to plot the radial roller follower motion. The plot is shown below:

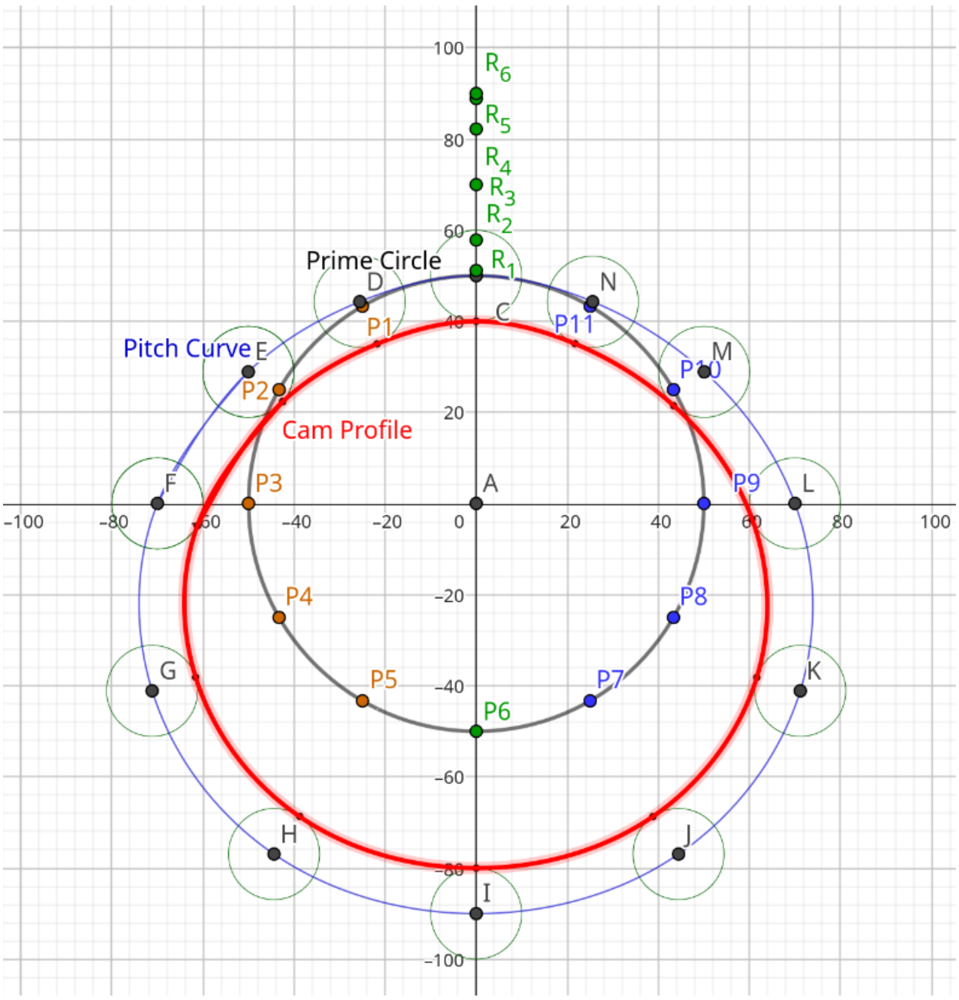

Figure 1: Radial Roller Follower Motion

1.1.1 Implementation

In order to make life a bit easier, I used `Sequence` function in Geogebra to easily plot the Cam Profile. The code is shown below:

$$f_{\text{rise}}(\theta) = 40\left(\frac{\theta}{180} - \frac{1}{2\,\pi}\sin\left(2\,\pi\,\frac{\theta}{180}\right)\right)$$

$$f_{\text{return}}(\theta) = 40 - 40\left(\frac{\theta - 180}{180} - \frac{1}{2\,\pi}\sin\left(2\,\pi\,\frac{\theta - 180}{180}\right)\right)$$

r = 50

-5 50

Figure 2: Putting equationsin Geogebra

$\text{cond} = \text{If}(\theta < 180°, 0, 1)$

$= 0$

$R = \text{If}\left(\text{cond} < 0.5, f_{\text{rise}}\left(\frac{\theta}{1°}\right), f_{\text{return}}\left(\frac{\theta}{1°}\right)\right)$

$= 0$

Figure 3: Using `If` condition for piecewise function

points = {E, F, G, H, I, J, K, L, M, N, C, D, E, F}

= {(-50.07, 28.91), (-70, 0), (-71.17, -41.09), (-44.42, -76.94), (0, -90), (44.42, -76

a = Spline(points, 3)

$= \begin{array}{l} x = \text{If}(t < 0.07, 8904.97\,t^3 - 320.01\,t - 50.07, t < 0.16, -558.78\,t^3 + 2077 \\ y = \text{If}(t < 0.07, -4435.43\,t^3 - 371.38\,t + 28.91, t < 0.16, 6312.5\,t^3 - 2359 \end{array}$

$r_{\text{roller}} = 10$

-5 20

NO = 14

1 14

circles = Sequence(Circle(points(k), r_{roller}), k, 1, NO, 1)

= {$(x + 50.07)^2 + (y - 28.91)^2 = 100$, $(x + 70)^2 + y^2 = 100$, $(x + 71.17)^2 + (y +$

tangents = Sequence(Tangent(points(k), a), k, 1, NO, 1)

= {y = 1.16x + 87.02, y = 1.8x + 125.76, y = -3.23x - 271.09, y = -0.68x - 107.2

normals = Sequence(PerpendicularLine(points(k), tangents(k)), k, 1, NO, 1)

= {320.01x + 371.38y = -5287.66, 237.98x + 427.56y = -16658.58, -145.57x + 47

intersects = Sequence(Intersect(circles(k), normals(k), 2), k, 1, NO, 1)

= {(-42.5, 22.38), (-61.26, -4.86), (-61.62, -38.13), (-38.79, -68.68), (0.01, -80), (3

b = Spline(intersects, 3)

$= \begin{array}{l} x = \text{If}(t < 0.08, 7461.53\,t^3 - 283.14\,t - 42.5, t < 0.16, -820.45\,t^3 + 1975. \\ y = \text{If}(t < 0.08, -3494.89\,t^3 - 320.5\,t + 22.38, t < 0.16, 5866.8\,t^3 - 2233. \end{array}$

Figure 4: Using `Sequence` function to plot the cam profile

2 Radial Roller: Analytical Method

The Analytical equations for the radial roller follower motion is given by:

$$X_C = \{R_r \sin(\phi)\} \cos(\theta) + (R_P + y(\theta) - R_r \cos(\phi)) \sin(\theta)$$
$$Y_C = -\{R_r \sin(\phi)\} \sin(\theta) + (R_P + y(\theta) - R_r \cos(\phi)) \cos(\theta)$$

And the pressure angle ϕ is given by:

$$\phi = \tan^{-1}\left\{\frac{y'(\theta)}{R_P + y(\theta)}\right\}$$

The Cam Profile is shown below:

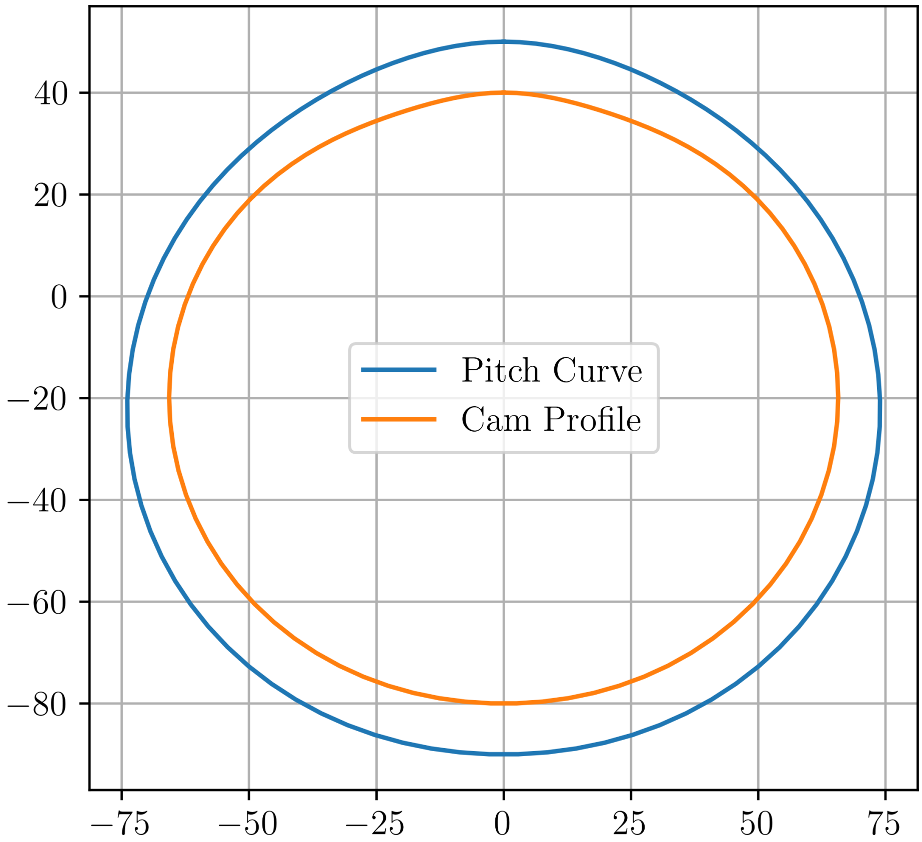

Figure 5: Radial Roller Follower Motion($R_P = 50$ mm)

The pressure angle variation is shown below:

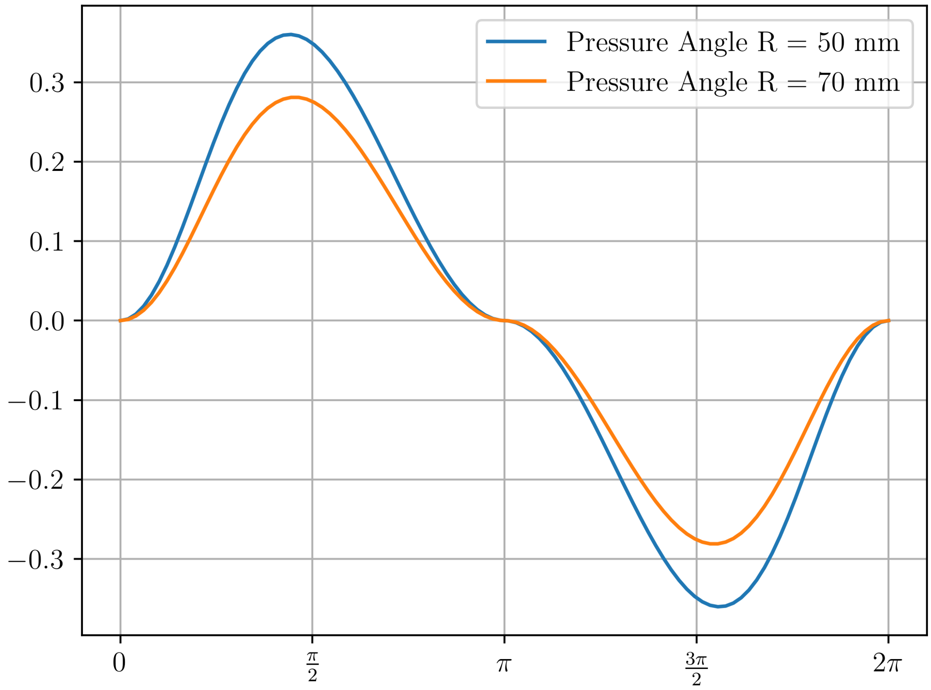

Figure 6: Pressure Angle Variation

3 Flat Faced Follower: Analytical

The Analytical equations for the flat faced follower motion is given by:

$$X_C(\theta) = y'(\theta)\cos(\theta) + (R_b + y(\theta))\sin(\theta)$$
$$Y_C(\theta) = -y'(\theta)\sin(\theta) + (R_b + y(\theta))\cos(\theta)$$

The Cam Profile is shown below:

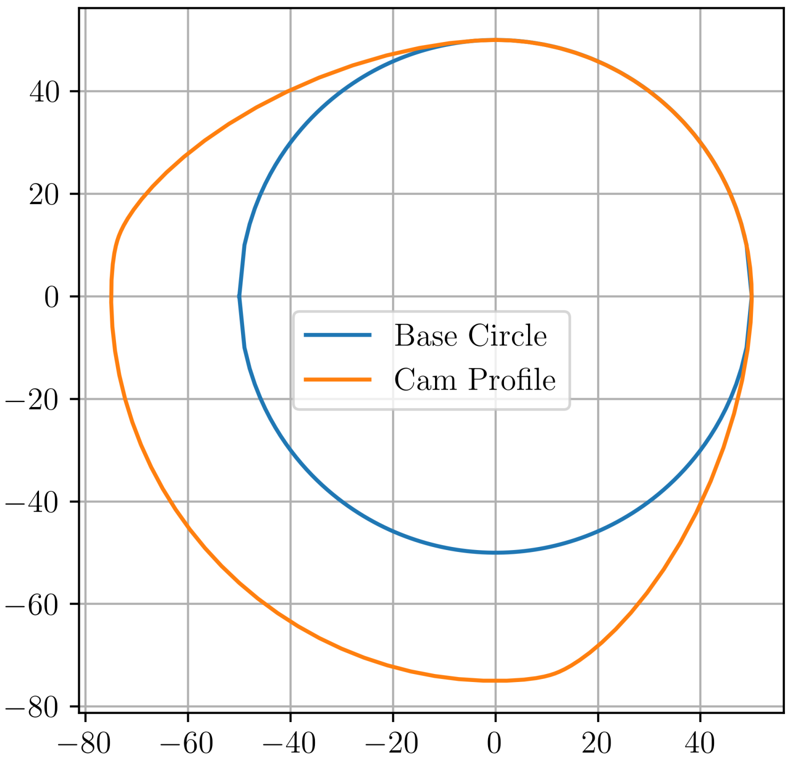

Figure 7: Flat Faced Follower Motion

The minimum width of the flat faced follower is given by:

$$\text{Minimum Width} = y'_{\text{max}} + y'_{\text{min}}$$

This value is calculated to be **59.6825 mm**.

4 Code

I have used Python to plot the cam profile and pressure angle variation analytically. The code given below is for both Q2 and Q3.

```
# %%
import sympy as sp
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
from sympy import pi

# %%
t = sp.symbols("t")
PRECISION = 100


class Range:
    def __init__(self, start, end):
        self.start, self.end = start, end

    def length(self):
        return self.end - self.start


class Interval:
    def __init__(self, value, t_range, y_range):
        if value in ("345", "3-4-5"):
            value = self._three_four_five(t_range, y_range)
        elif value in ("cycloid", "cycloidal"):
            value = self._cycloid(t_range, y_range)
        else:
            value = sp.sympify(value)
        self.value = value

    def _three_four_five(self, t_range: Range, y_range: Range):
        t = sp.symbols("t")
        var = (t - t_range.start) / t_range.length()
        expr = 10 * var**3 - 15 * var**4 + 6 * var**5
        return y_range.start + y_range.length() * expr

    def _cycloid(self, t_range: Range, y_range: Range):
        var = (t - t_range.start) / t_range.length()
        expr = var - (sp.sin(2 * pi * var) / (2 * pi))
        return y_range.start + y_range.length() * expr

    def __call__(self, t):
        return self.value.subs("t", t)
```

```python
class Motion:
    def __init__(self, *args):
        y = []
        for value, t_range, y_range in args:
            interval = Interval(value, t_range, y_range)
            y.append((interval, t_range))
        self.y = sp.Piecewise(
            *[
                (interval.value, sp.And(t_range.start <= t, t < t_range.end))
                for interval, t_range in y
            ]
        )

    def __call__(self, t):
        return self.y.subs("t", t)

    def plot_rise(self, ax, **kwargs):
        x = np.linspace(0, 2 * np.pi, PRECISION)
        y = [self.y.subs("t", t) for t in x]
        ax.plot(x, y, **kwargs)

    def plot_velocity(self, ax, **kwargs):
        x = np.linspace(0, 2 * np.pi, PRECISION)
        y_dot = sp.diff(self.y, t)
        y = [y_dot.subs("t", t) for t in x]
        ax.plot(x, y, **kwargs)

    def plot_acceleration(self, ax, **kwargs):
        x = np.linspace(0, 2 * np.pi, PRECISION)
        y_dot = sp.diff(self.y, t)
        y_dot_dot = sp.diff(y_dot, t)
        y = [y_dot_dot.subs("t", t) for t in x]
        ax.plot(x, y, **kwargs)


class Cam:
    def __init__(self, radius, e, direction):
        self.R = radius
        self.e = e
        if direction in ("ACW", "CCW", "anticlockwise"):
            self.direction = 1
        elif direction in ("CW", "clockwise"):
            self.direction = -1

    def plot_base_circle(self, ax, **kwargs):
        x = np.linspace(-self.R, self.R, PRECISION)
        y = np.sqrt(self.R**2 - x**2)
        x = np.concatenate((x, x[::-1]))
        y = np.concatenate((y, -y[::-1]))
        ax.plot(x, y, **kwargs)

    def pressure_angle(self, motion):
        return sp.atan(
            (sp.diff(motion.y, t) - self.e)
            / (sp.sqrt(self.R**2 - self.e**2) + motion.y)
```

```
        )

    def plot_pressure_angle(self, ax, motion, **kwargs):
        phi = self.pressure_angle(motion)
        x_phi, y_phi = [], []
        for t in np.linspace(0, 2 * np.pi, PRECISION):
            x_phi.append(t)
            y_phi.append(phi.subs("t", t))

        x_ticks = kwargs.pop("x_ticks", [0, 0.5 * np.pi, np.pi, 1.5 * np.pi,
           2 * np.pi])
        x_ticklabels = kwargs.pop(
            "x_ticklabels",
            ["$0$", r"$\frac{\pi}{2}$", r"$\pi$", r"$\frac{3\pi}{2}$", r"$2\
               pi$"],
        )
        ax.set_xticks(x_ticks)
        ax.set_xticklabels(x_ticklabels)

        ax.plot(x_phi, y_phi, **kwargs)

    def pitch_circle(self, motion):
        X_p = (self.R + motion.y) * sp.sin(t) + self.e * sp.cos(t)
        Y_p = (self.R + motion.y) * sp.cos(t) - self.e * sp.sin(t)
        return X_p * self.direction, Y_p

    def plot_pitch_circle(self, ax, motion, **kwargs):
        X_p, Y_p = self.pitch_circle(motion)
        x_p, y_p = [], []
        for t in np.linspace(0, 2 * np.pi, PRECISION):
            x_p.append(X_p.subs("t", t))
            y_p.append(Y_p.subs("t", t))
        ax.plot(x_p, y_p, **kwargs)


class Follower:
    pass


class Roller(Follower):
    def __init__(self, radius):
        self.R = radius

    def cam_profile(roller, motion, cam):
        phi = cam.pressure_angle(motion)
        X_r = -roller.R * sp.sin(phi) * sp.cos(t) + (
            cam.R + motion.y - roller.R * sp.cos(phi)
        ) * sp.sin(t)
        Y_r = roller.R * sp.sin(phi) * sp.sin(t) + (
            cam.R + motion.y - roller.R * sp.cos(phi)
        ) * sp.cos(t)
        return X_r * cam.direction, Y_r

    def plot_cam_profile(self, ax, motion, cam, **kwargs):
        X_r, Y_r = self.cam_profile(motion, cam)
```

```
        x_r, y_r = [], []
        for t in np.linspace(0, 2 * np.pi, PRECISION):
            x_r.append(X_r.subs("t", t))
            y_r.append(Y_r.subs("t", t))
        ax.plot(x_r, y_r, **kwargs)


class Flat_Face(Follower):
    def __init__(self):
        pass

    def cam_profile(roller, motion, cam):
        X_c = (cam.R + motion.y) * sp.sin(t) + sp.diff(motion.y, t) * sp.cos(
            t)
        Y_c = (cam.R + motion.y) * sp.cos(t) - sp.diff(motion.y, t) * sp.sin(
            t)
        return X_c * cam.direction, Y_c

    def plot_cam_profile(self, ax, motion, cam, **kwargs):
        X_c, Y_c = self.cam_profile(motion, cam)
        x_c, y_c = [], []
        for t in np.linspace(0, 2 * np.pi, PRECISION):
            x_c.append(X_c.subs("t", t))
            y_c.append(Y_c.subs("t", t))
        ax.plot(x_c, y_c, **kwargs)

    def min_width(self, motion):
        max_y_dot = 0
        min_y_dot = 0
        y_dot = sp.diff(motion.y, t)
        for t_val in np.linspace(0, 2 * np.pi, PRECISION * 10):
            max_y_dot = max(max_y_dot, y_dot.subs("t", t_val))
            min_y_dot = min(min_y_dot, y_dot.subs("t", t_val))

        return (max_y_dot + abs(min_y_dot)).evalf()

fig, ax = plt.subplots()
ax.set_aspect("equal")

steps = [
    ("cycloidal", Range(0, pi), Range(0, 40)),
    ("cycloidal", Range(pi, 2 * pi), Range(40, 0)),
]
motion = Motion(*steps)

cam = Cam(radius=50, e=0, direction="CW")
cam.plot_pitch_circle(ax, motion, label="Pitch Curve")
follower = Roller(radius=10)
follower.plot_cam_profile(ax, motion, cam, label="Cam Profile")

plt.legend()
plt.grid()
plt.show()
plt.close()
```

```python
fig, ax = plt.subplots()
ax.set_aspect("equal")

steps = [
    (0, Range(0, pi / 2), Range(0, 0)),
    ("345", Range(pi / 2, pi), Range(0, 25)),
    (25, Range(pi, 3 * pi / 2), Range(25, 25)),
    ("345", Range(3 * pi / 2, 2 * pi), Range(25, 0)),
]
motion = Motion(*steps)
cam = Cam(radius=50, e=15, direction="ACW")
cam.plot_base_circle(ax, label="Base Circle")
follower = Flat_Face()
follower.plot_cam_profile(ax, motion, cam, label="Cam Profile")
print("Minimum Width:", follower.min_width(motion))

plt.legend()
plt.grid()
plt.show()
plt.close()
```