—Python coding for generating Spur gear—

Contact Dr. Khalif H. Harib (k.harib@uaeu.ac.ae) or Prof. Sharif Ullah (ullah@mail.kitami-it.ac.jp), if you have any query.

class Gear:

 def \_\_init\_\_(self, color, module, Nt):

 self.nt = Nt

 addendum = module

 dedundum = 1.157 \* module

 Rp = module \* Nt / 2 # pitch circle radius

 Rb = Rp \* np.cos(alpha) # base circle radius

 Rd = Rp - dedundum # dedundum circle radius

 Ra = Rp + addendum # adendum circle radius

 CT = 2 \* Rp \* np.sin(PI / 2 / Nt) # circular thickness

 #

 # define phi as the angle between a vertical axis with the radial line to the pitch circle intersection with the profile

 sinphi = CT / 2 / Rp

 cosphi = np.sqrt(1 - sinphi \* sinphi)

 xp = CT / 2

 yp = Rp \* cosphi

 #

 # to find theta\_b and theta\_p, we use the Newton-Raphson method to solve 2 equations iteratively

 theta\_b = 0.5 # initial guess

 theta\_p = 1.0 # initial guess

 qx = - Rb \* np.sin(theta\_p) + Rb \* (theta\_b + theta\_p) \* np.cos(theta\_p)

 qy = Rb \* np.cos(theta\_p) + Rb \* (theta\_b + theta\_p) \* np.sin(theta\_p)

 f = np.array([xp - qx, yp - qy])

 while np.sqrt(f[0] \* f[0] + f[1] \* f[1]) > TOL: # iterate

 Jxb = - Rb \* np.cos(theta\_p)

 Jyb = - Rb \* np.sin(theta\_p)

 Jxp = Rb \* np.cos(theta\_p) + Rb\*(theta\_b + theta\_p) \* np.sin(theta\_p) - Rb \* np.cos(theta\_p)

 Jyp = Rb \* np.sin(theta\_p) - Rb\*(theta\_b + theta\_p) \* np.cos(theta\_p) - Rb \* np.sin(theta\_p)

 J = np.array([[Jxb, Jxp],[Jyb, Jyp]])

 invJ = np.linalg.inv(J)

 Correction = np.matmul(invJ, f)

 theta\_b = theta\_b - Correction[0]

 theta\_p = theta\_p - Correction[1]

 qx = - Rb \* np.sin(theta\_p) + Rb \* (theta\_b + theta\_p) \* np.cos(theta\_p)

 qy = Rb \* np.cos(theta\_p) + Rb \* (theta\_b + theta\_p) \* np.sin(theta\_p)

 f[0] = xp - qx

 f[1] = yp - qy

 #

 # to find theta\_a: solve one equation

 theta\_a = math.sqrt((Ra \* Ra - Rb \* Rb) / Rb / Rb) - theta\_b

 #

 # To find the psi (fillet angle): solve 2 equations simultaneously

 xB = Rb \* np.sin(theta\_b)

 yB = Rb \* np.cos(theta\_b)

 Rdr\_sqr = r \* r + (Rd + r) \* (Rd +r)

 sinpsi = (xB \* (Rd+ r) + yB \* r) / Rdr\_sqr

 cospsi = (xB \* r - yB \* (Rd+ r)) / Rdr\_sqr

 psi = math.atan2(sinpsi, cospsi)

 xE = (Rd + r) \* np.sin(PI - psi)

 yE = (Rd + r) \* np.cos(PI - psi)

 #

 # Generate the points for the profile of the first tooth starting with the fillet

 self.points = []

 theta = PI / 2 + psi

 theta\_inc = PI / 2 / Nfp

 for i in range(Nfp): # generate the root fillet

 x = xE + r \* np.cos(theta)

 y = yE + r \* np.sin(theta)

 self.points.append(x)

 self.points.append(y)

 theta = theta - theta\_inc

 theta\_inc = (theta\_a + theta\_b) / Nep

 theta = - theta\_b + theta\_inc

 for i in range(Nep): # generate the involute curve

 qx = - Rb \* np.sin(theta) + Rb \* (theta\_b + theta) \* np.cos(theta)

 qy = Rb \* np.cos(theta) + Rb \* (theta\_b + theta) \* np.sin(theta)

 self.points.append(qx)

 self.points.append(qy)

 theta = theta + theta\_inc

 for j in range(Nep + Nfp): # generate the left-hand side of the tooth

 i = Nep + Nfp - 1 - j

 self.points.append(-self.points[2\*i])

 self.points.append(self.points[2\*i+1])

 #

 # Use the coordinates of the points of the generated tooth profile after rotating them by an angle incremented by 2PI/Nt

 theta = 0

 self.allpoints = []

 for j in range(Nt):

 for i in range(2 \* (Nep + Nfp)):

 self.allpoints.append(self.points[2\*i] \* np.cos(theta) - self.points[2\*i+1] \* np.sin(theta))

 self.allpoints.append(self.points[2\*i] \* np.sin(theta) + self.points[2\*i+1] \* np.cos(theta))

 theta = theta + 2 \* PI / Nt

 self.allpoints.append(self.points[0]) # use the first point coordinates for the last point

 self.allpoints.append(self.points[1])

 #

 # Create the shapes

 self.gearshape = canvas.create\_polygon(self.allpoints, fill = color, width = 1)

 self.crooshair\_h = canvas.create\_line(-10, 0, 10, 0)

 self.crooshair\_v = canvas.create\_line(0, -10, 0, 10)

 print("Rp = ", Rp)

 #

 # The function to place and orient the gear

 def rotate\_and\_translate(self, theta, xdisp, ydisp):

 self.newpoints = []

 for i in range (self.nt \* 2 \* (Nep + Nfp) + 1):

 self.newpoints.append(xdisp + self.allpoints[2\*i] \* np.cos(theta) - self.allpoints[2\*i+1] \* np.sin(theta))

 self.newpoints.append(ydisp + self.allpoints[2\*i] \* np.sin(theta) + self.allpoints[2\*i+1] \* np.cos(theta))

 canvas.coords(self.gearshape, self.newpoints)

 canvas.coords(self.crooshair\_h, xdisp-10, ydisp, xdisp+10, ydisp)

 canvas.coords(self.crooshair\_v, xdisp, ydisp-10, xdisp, ydisp+10)

The End