① Homogeneous Transformations

1.1 Composition of transformation

Given world axis fixed

① Rotate by ϕ about world x-axis

② Translate by y along the current y-axis

③ Rotate by θ about the world z-axis

④ Rotate by ψ about the current x-axis

Rangle → Zero translation (4×4) I is Identity matrix (4×4)

Transform (with identity Rotation) (4×4)

R_ϕ → Rotation by ϕ about - world axis

T_y → Translation by y along current y-axis

R_θ → Rotation by θ about world z-axis

R_ψ → Rotate by ψ about current x-axis

$$T = R_\theta R_\phi I T_y R_\psi$$ (Reason for this order) ↓

* We pre-multiply transformation about fixed frame and post multiply transformations about current frame

```
from sympy import *
init_printing()


phi,y,theta,psi=symbols('phi, y, theta, psi')

"""
R_phi,T_y,R_theta,R_psi=symbols('R_phi,T_y,R_theta,R_psi')
"""

R_phi=Matrix(
    [[1,0,0,0],
    [0,cos(phi),-sin(phi),0],
    [0,sin(theta),cos(theta),0],
    [0,0,0,1]])


T_y=Matrix([[1,0,0,0],
            [0,1,0,y],
            [0,0,1,0],
            [0,0,0,1]])



R_theta=Matrix(
    [[cos(theta),-sin(theta),0,0],
    [sin(theta),cos(theta),0,0],
    [0,0,1,0],
    [0,0,0,1]])



R_psi=Matrix(
    [[1,0,0,0],
    [0,cos(psi),-sin(psi),0],
    [0,sin(psi),cos(psi),0],
    [0,0,0,1]])


#This is the final MAtrix Required
expr=simplify(R_theta*R_phi*T_y*R_psi)
pprint(expr)
```

```
shivam@shivam-MacBookAir:~/Modelling_662$ /bin/python /home/shivam/Modelling_662/HW2_codes/question1_1.py
⎡cos(θ)  -sin(θ)·cos(φ + ψ)  sin(θ)·sin(φ + ψ)   -y·sin(θ)·cos(φ)⎤
⎢                                                               ⎥
⎢sin(θ)  cos(θ)·cos(φ + ψ)   -sin(φ + ψ)·cos(θ)  y·cos(φ)·cos(θ)⎥
⎢                                                               ⎥
⎢  0         sin(ψ + θ)          cos(ψ + θ)          y·sin(θ)   ⎥
⎢                                                               ⎥
⎣  0           0                   0                   1       ⎦
```

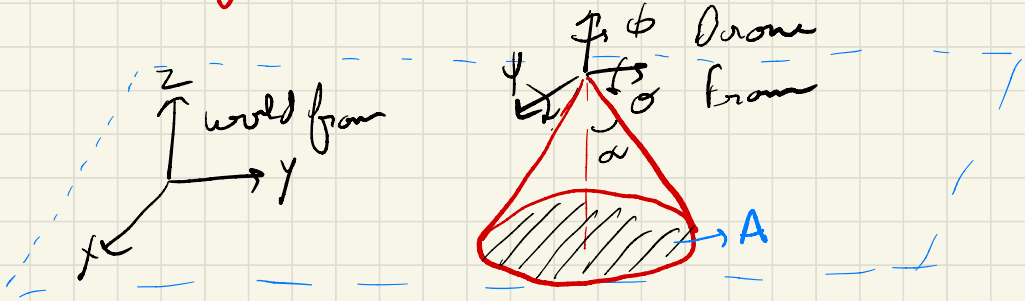
(2) Modeling beyond rigid transformations

Cone angle $\alpha = 45^\circ$

Coverage area $A = A$

3 consecutive transformations (ψ, θ, ϕ)

location of Drone (dx, dy, dz)



Transformation from

Drone frame to world frame

$R_\psi \rightarrow$ Rotation about x

$R_\theta \rightarrow$ Rotation about y

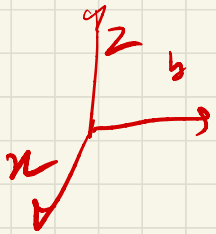
$R_\phi \rightarrow$ Rotation about z

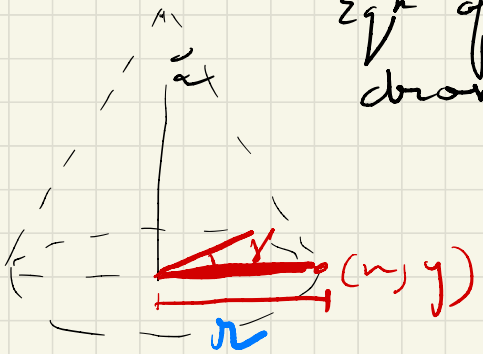
(ψ, θ, ϕ) Consecutive

$$T_d = \left[\begin{array}{c|c} I & \begin{matrix} dx \\ dy \\ dz \end{matrix} \\ \hline 0\ 0\ 0 & 1 \end{array}\right]$$

$$T_D^0 = T_d R_\phi R_\theta R_\psi$$

Eqn of circle wrt drone





For any point x,y on the circle let the $\angle = \gamma$

and radius of circle is r

$\therefore x = r \sin\gamma$

$y = r \cos\gamma$

$x^2 + y^2 = r^2(\cos^2\gamma + \sin^2\gamma)$

$x^2 + y^2 = r^2$

$x^2 + y^2 = z^2 \tan^2\alpha$

$\alpha = 45$

$\therefore \boxed{x^2 + y^2 = z^2}$ — Eqn of cone wrt drone

From the cone we know that

$\tan\alpha = \frac{r}{z}$

$\therefore r = z\tan\alpha$

Now we convert the eqn of cone from drone frame to ground frame and then equate with the x, y plane.

if $P_D = [x, y, z, 1]^T$

the eq^{n} of cone can be written as

$$\therefore P_D^T \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} P_D$$

with respect to drone

$$= P_D^T S P_D$$

To change the frame of cone to origin we do the following

$= (T_0^0 P_0)^T S \; T_0^0 P_D$ (Transforming Pa to ground frame)

$= P_0^T \underbrace{T_0^{0T} S \; T_0^0}_{H} P_D$

Eqn of cone in ● frame is $= P_0^T H P_0$

Computed in the code

Now this cone intersects with the x-y plane to get the eqn of an ellipse

$\therefore$ $\boxed{I_n P_0^T H P_0}$

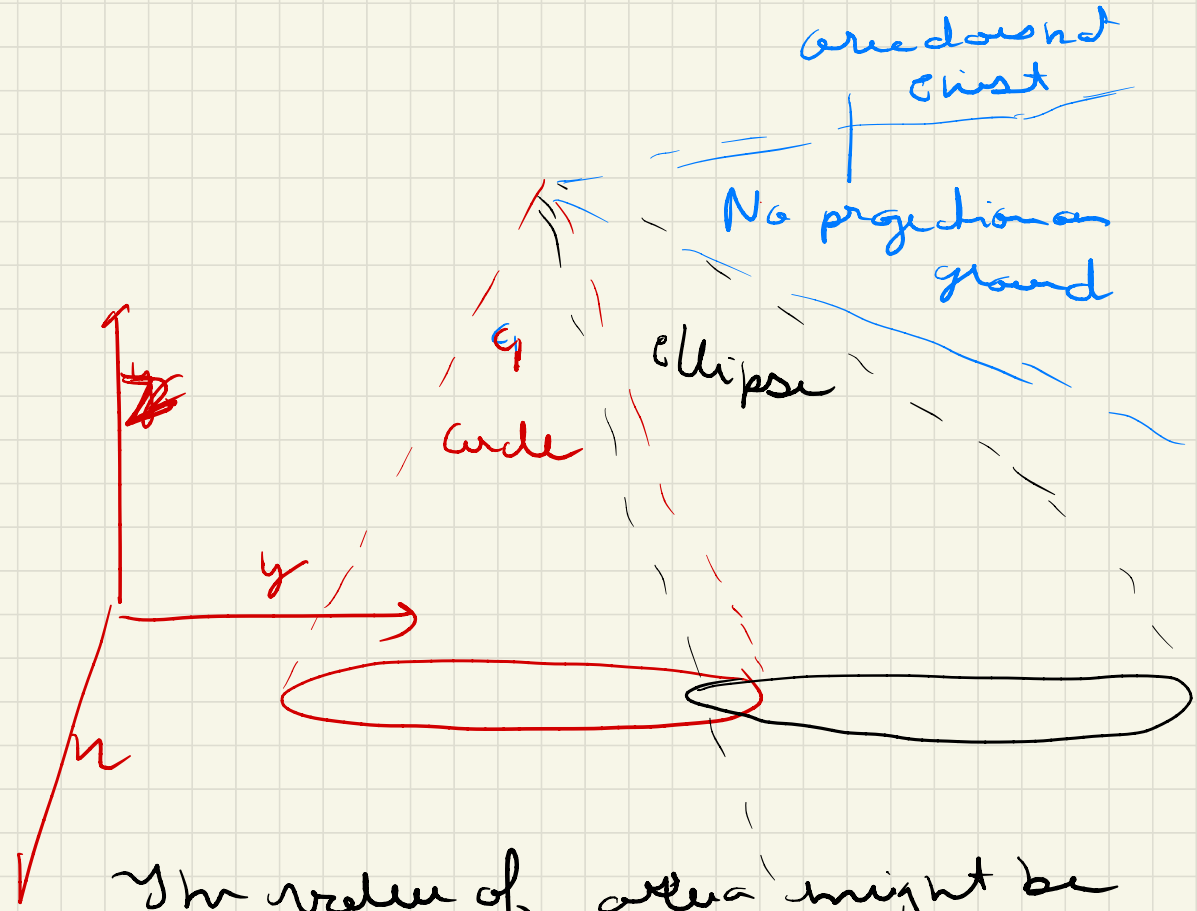
put $z=0$

we get eqn of form

$ax^2 + 2bxy + cy^2 + dx + ey + f = 0$

Area of ellipse

$$= \frac{-\pi}{(ac-b^2)^{3/2}} \begin{vmatrix} a & b & d \\ b & c & e \\ d & e & f \end{vmatrix}$$



The values of area might be complex for some values of θ, ϕ, ψ as for those values no area exists. There might be no intersection as shown above

```
#First rotation matrix are written about each axis
#Then a transformation matrix is found out for the rotations
#Then equation of cone is written in matrix format wrt drone.
#Then the equation of the cone is converted to the origin frame.
#Then we substitute z=0 to find the intersection b/w the cone and the x-y plane.
#This gives equation of an elipse which is then solved for area using the hint given.



# Sympy is imported to make it possible to write equations
from sympy import *
init_printing()

#symbols used in the code are defined as follows
x,y,z,d_x,d_y,d_z=symbols('x,y,z,d_x,d_y,d_z')
phi,theta,psi=symbols('phi,theta, psi')


#Rotation matrix are found out for each rotation.
R_psi=Matrix(
    [[1,0,0,0],
     [0,cos(psi),-sin(psi),0],
     [0,sin(psi),cos(psi),0],
     [0,0,0,1]])

R_theta=Matrix(
    [[cos(theta),0,sin(theta),0],
     [0,1,0,0],
     [-sin(theta),0,cos(theta),0],
     [0,0,0,1]])
R_phi=Matrix(
    [[cos(phi),-sin(phi),0,0],
     [sin(phi),cos(phi),0,0],
     [0,0,1,0],
     [0,0,0,1]])
T_D_wrt_origin=Matrix(
    [[1,0,0,d_x],
     [0,1,0,d_y],
     [0,0,1,d_z],
     [0,0,0,1]])

#Transformation of Drone wrt origin is given by the below equation.
Rotation_Dnew_wrt_origin=simplify(T_D_wrt_origin*R_phi*R_theta*R_psi)

#any point wrt drone can be written as.
p_drone=Matrix([[x],[y],[z],[1]])

#S is a matrix to make the equation of the cone.
S=Matrix([[1,0,0,0],[0,1,0,0],[0,0,-1,0],[0,0,0,0]])


#Equation of the cone wrt the origin frame can be found using the below computation
#This is explaned in the pdf report also.
H=(Rotation_Dnew_wrt_origin.T)*S*(Rotation_Dnew_wrt_origin)
Equation_cone_wrt_origin=simplify(p_drone.T*H*p_drone)

#To find the intersection with the x-y plane we substitute z=0
Intersection_cone_x_y_plane=Equation_cone_wrt_origin.subs({z:0})

#To convert this final cone equation to
Ellipse_eqn=Intersection_cone_x_y_plane[0]
pprint(Ellipse_eqn)

#folowing steps are followed to find the coefficent of x,y(square) x*y, x, y, c
#Then these are stored as a list
#The a formula as given in hint is used to evaluate the final area.
expr_1=Poly(Ellipse_eqn.evalf(),x,y,x*y,x**2,y**2)

pprint(expr_1)
list1=expr_1.coeffs()

a=list1[0]
b=list1[1]/2
c=list1[3]
d=list1[2]/2
```

```
e=list1[4]/2
f=list1[5]

# Area from an elispse eqn is given by as follows.
k=(-pi)/(sqrt(((a*c)-(b*b))**3))
A=Matrix([[a,b,d],[b,c,e],[d,e,f]])
Det_A=(det(A))

#This is the area of the ellipse
area=(k*Det_A)
pprint(simplify(area))

#This is the area for particular values of variables
print(area.subs({psi:pi/6,theta:pi/6,phi:0,d_x:1,d_y:2,d_z:3}))
```

→ Value of Area for these values

$= -22.627 \times \pi \times (-1.75 - 0.625\sqrt{3})$

$= +22.627\pi(1.75 + 0.625\sqrt{3})$

For any value of $x=0$ $y=0$

$psi = 0$ $theta = 0$

the area comes out to be

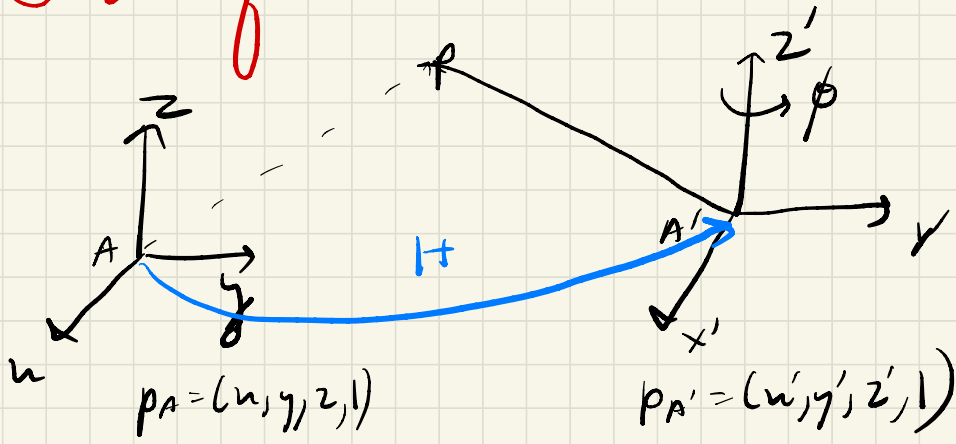
$= \pi z^2$

which should be the case as

the ellipse that the drone has just rotated about z and projecting a circle on x-y plane

$x^2 = z^2$ $\quad x = z$

(1.3) Transform Estimation



$p_A = (x, y, z, 1)$ $\quad p_{A'} = (x', y', z', 1)$

Given that :- A is only rotated about Z axis with angle ϕ and translated freely in space

$$H = T^A_{A'} = \begin{bmatrix} R^A_{A'} & t \\ 0 \;\; 0 \;\; 0 & 1 \end{bmatrix}$$

t = origin of A' w.r.t A

$t = (a, b, c)$

$\therefore$ for Point P

$$P_A = T^A_{A'} P_{A'}$$

$$R_{A'}^{A} = R_{\phi} = \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$T_{A'}^{A} = \begin{bmatrix} \cos\phi & -\sin\phi & 0 & a \\ \sin\phi & \cos\phi & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$P_A = T_{A'} P_{A'}$$

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\phi & -\sin\phi & 0 & a \\ \sin\phi & \cos\phi & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix}$$

$$x = x'\cos\phi - y'\sin\phi + 0 + a$$

$$y = x'\sin\phi + y'\cos\phi + 0 + b$$

$$z = 0 + 0 + z' + c$$

$\therefore$ we get

$$a = x - x'\cos\phi + y'\sin\phi$$

$$b = y - x'\sin\phi - y'\cos\phi$$

$$c = z - z'$$

$$H_{A'}^{A} = \begin{bmatrix} \cos\phi & -\sin\phi & 0 & x - x'\cos\phi + y'\sin\phi \\ \sin\phi & \cos\phi & 0 & y - x'\sin\phi - y'\cos\phi \\ 0 & 0 & 1 & z - z' \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$H_{A}^{A'} = \left(H_{A'}^{A}\right)^{-1} = \begin{bmatrix} R^T & -R^T d \\ 0 & 1 \end{bmatrix}$$

```
from sympy import *
init_printing()

#Defining the symbols used in the code
x,y,z=symbols('x,y,z')
a,b,c=symbols('a,b,c')
phi=symbols('phi')
xprime,yprime,zprime=symbols('xprime,yprime,zprime')

"""        Origin of frame A wrt to frame A'   x_A,y_A,z_A  """
x_A,y_A,z_A=symbols('x_A,y_A,z_A')

"""     Transformation of A with respect of A'
        pre multiplying this matrix by points of A will give points of A wrt to A'

"""
# any point wrt A frame
p_A=Matrix([[x],[y],[z],[1]])

# Any point wrt A_prime frame
p_Aprime=Matrix([[xprime],[yprime],[zprime],[1]])

#Tansfromation matrix from Aprime to A
T_Aprime_wrt_A=Matrix([[cos(phi),-sin(phi),0,x_A],[sin(phi),cos(phi),0,y_A],[0,0,1,z_A],[0,0,0,1]])


#Therefore pA= T_Aprime_wrt_A*p_Aprime
Product=simplify(T_Aprime_wrt_A*p_Aprime)
pprint(Product)

# origin of Aprime with respect to A
o_Aprime=solve([Product-p_A],[x_A,y_A,z_A])

#Final Matrix with the origin values put in
T_Aprime_wrt_A=Matrix([[cos(phi),-sin(phi),0,o_Aprime[x_A]],[sin(phi),cos(phi),0,o_Aprime[y_A]],[0,0,1,o_Aprime[z_A]],[0,0,0,1]])
pprint(T_Aprime_wrt_A)


#But we have been given the question to find T_A_wrt_Aprime
#For that I have done the following computations explained int he PDF report.
R_Aprime_wrt_A=Matrix([[cos(phi),-sin(phi),0],[sin(phi),cos(phi),0],[0,0,1]])
Origin=Matrix([[o_Aprime[x_A]],[o_Aprime[y_A]],[o_Aprime[z_A]]])

R_A_wrt_Aprime=R_Aprime_wrt_A.T
Origin_new=-(R_A_wrt_Aprime*Origin)


#This the final matrix required.
T_Aprime_wrt_A=Matrix([[R_A_wrt_Aprime[0,0],R_A_wrt_Aprime[0,1],R_A_wrt_Aprime[0,2],Origin_new[0]],
                       [R_A_wrt_Aprime[1,0],R_A_wrt_Aprime[1,1],R_A_wrt_Aprime[1,2],Origin_new[0]],
                       [R_A_wrt_Aprime[2,0],R_A_wrt_Aprime[2,1],R_A_wrt_Aprime[2,2],Origin_new[0]],
                       [0,0,0,1]])

pprint(T_Aprime_wrt_A)
```

Python Outputs

```
shivam@shivam-MacBookAir:~/Modelling_662$ /bin/python /home/shivam/Modelling_662/HW2_codes/questio1_3.py
⎡x_A + x′·cos(φ) - y′·sin(φ)⎤
⎢                           ⎥
⎢x′·sin(φ) + y_A + y′·cos(φ)⎥
⎢                           ⎥
⎢        z_A + z′           ⎥
⎢                           ⎥
⎣           1               ⎦
⎡cos(φ)  -sin(φ)  0  x - x′·cos(φ) + y′·sin(φ)⎤
⎢                                             ⎥
⎢sin(φ)  cos(φ)   0  -x′·sin(φ) + y - y′·cos(φ)⎥
⎢                                             ⎥
⎢  0       0      1          z - z′           ⎥
⎢                                             ⎥
⎣  0       0      0            1              ⎦
⎡cos(φ)   sin(φ)  0  -(x - x′·cos(φ) + y′·sin(φ))·cos(φ) - (-x′·sin(φ) + y - y′·cos(φ))·sin(φ)⎤
⎢                                                                                             ⎥
⎢-sin(φ)  cos(φ)  0  -(x - x′·cos(φ) + y′·sin(φ))·cos(φ) - (-x′·sin(φ) + y - y′·cos(φ))·sin(φ)⎥
⎢                                                                                             ⎥
⎢   0       0     1  -(x - x′·cos(φ) + y′·sin(φ))·cos(φ) - (-x′·sin(φ) + y - y′·cos(φ))·sin(φ)⎥
⎢                                                                                             ⎥
⎣   0       0     0                                  1                                       ⎦
shivam@shivam-MacBookAir:~/Modelling_662$
```

$T_{A'}^{A}$

$T_{A}^{A'}$

$C = H$

(2.1) Trajectory Optimisation

Drone moves from position x, y, z to x', y', z'

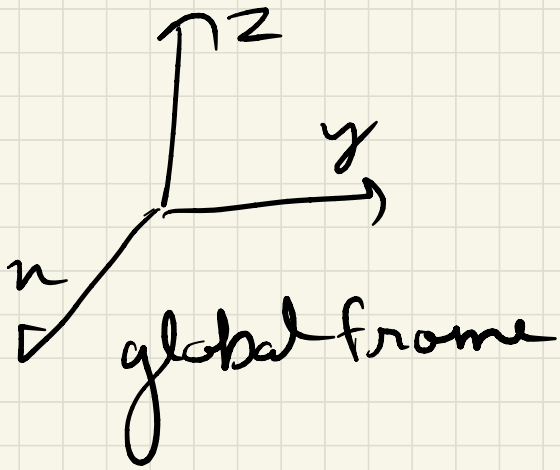
$\psi_g = 35°$, $\theta_g = 15°$, $\phi_g = 20°$

$w_{max} = 1 deg/s$

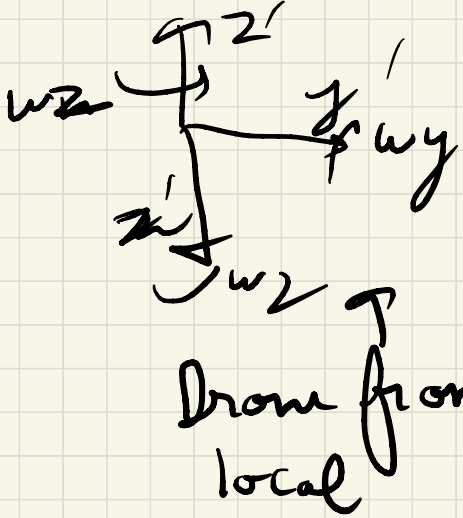
To plot trajectory of the drone such that it reaches the final orientation in the shortest time.

$$R_{final} = R_\phi R_\theta R_\psi$$

w_x, w_y, w_z are angular velocities of drone about the local x, y, z frame



global frame



Drone frame
local

Using the angle axis rotation we can find the axis $\hat{n}$ & the angle γ for the rotation of the frame.

$R(\hat{n}, \gamma)$

$|w_x| < 1$

$|w_y| < 1$

$|w_z| < 1$

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

$R(\gamma, \hat{n})$

$$\gamma = \cos^{-1}\left(\frac{Tr(R) - 1}{2}\right)$$

$$\gamma = \cos^{-1}\left(\frac{r_{11} + r_{22} + r_{33} - 1}{2}\right)$$

$$\gamma = \cos^{-1}\left(\frac{\cos\phi\cos\theta + \sin\phi\sin\psi\sin\theta + \cos\phi\cos\psi + \cos\psi\cos\theta - 1}{2}\right)$$

γ for $\psi = 35^\circ$ $\theta = 15$ $\phi = 20$

weight

$\gamma = 0.7079$ rads

or $\gamma = 40.55$ deg

$$\hat{n} = \frac{1}{2\sin\theta}\begin{bmatrix} R_{32} - R_{23} \\ R_{13} - R_{31} \\ R_{21} - R_{12} \end{bmatrix}$$

Using python to compute $\hat{n}$ for the given values

$$\hat{n} = \begin{bmatrix} 0.7847 \\ 0.5030 \\ 0.3621 \end{bmatrix}$$

$\omega_g = \hat{n}\dot{\gamma}$ → angular velocity about the $\hat{n}$ axis

$$\omega_g = \begin{bmatrix} 0.7847\,\dot{\gamma} \\ 0.5030\,\dot{\gamma} \\ 0.3621\,\dot{\gamma} \end{bmatrix}$$

$w_0 = \begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix}$ angular velocity of Drone in Local frame

$R(t) = [r_1(t) \quad r_2(t) \quad r_3(t)]$

$\dot{R}(t) = [\dot{r}_1 \quad \dot{r}_2 \quad \dot{r}_3]$

$= [w_g \times r_1 \quad w_g \times r_2 \quad w_g \times r_3]$

$= w_g \times R$

$= [w_g] R$ ← skew symmetric representation for cross product

$\dot{R} = w \times R = [w] R = [w]$

$[w_g] R = \dot{R} \rightarrow [w_g] R R^{-1} = \dot{R} R^{-1}$

$\rightarrow [w_g] = \dot{R} R^{-1} = \dot{R} R^T$

$$w_g = R_{gd}\, w_d$$

$$w_d = (R_{gd})^{-1} w_g = R^{-1} w_g$$

$$= R^T w_g$$

$$[w_d] = [R^T w_g] = R^T [w_g] R$$

Equation this

$$R^T [w_g] R$$

$$\boxed{[w_d] = R^T [w_g] R}$$

$$\begin{bmatrix} 0 & -w_z & w_y \\ w_z & 0 & -w_x \\ -w_y & w_x & 0 \end{bmatrix} = R^T \begin{bmatrix} 0 & -0.36\dot{\gamma} & 0.50\dot{\gamma} \\ 0.36\dot{\gamma} & 0 & -0.78\dot{\gamma} \\ -0.50\dot{\gamma} & 0.78\dot{\gamma} & 0 \end{bmatrix} R$$

$$[w_g] =$$

$$w_z$$

Solving these matrix we get

$$w_z = 0.3621\gamma$$

$$w_y = 0.5030\gamma$$

$$w_x = 0.7847\gamma$$

w_x is max out of these 3 and $|w_x|, |w_y|, |w_z| < 1$ deg/sec

$$\therefore w_x = 0.7847\gamma = 1$$

for min time

$$\therefore \gamma = \frac{1}{0.7847} = \boxed{1.2743}$$

$\gamma = 1.2743$ Deg/sec

$\gamma = 40.55 \text{ deg}$

$\therefore t = \frac{\gamma}{\dot{\gamma}} = \frac{40.55}{1.2743} = 31.82 \text{ sec}$

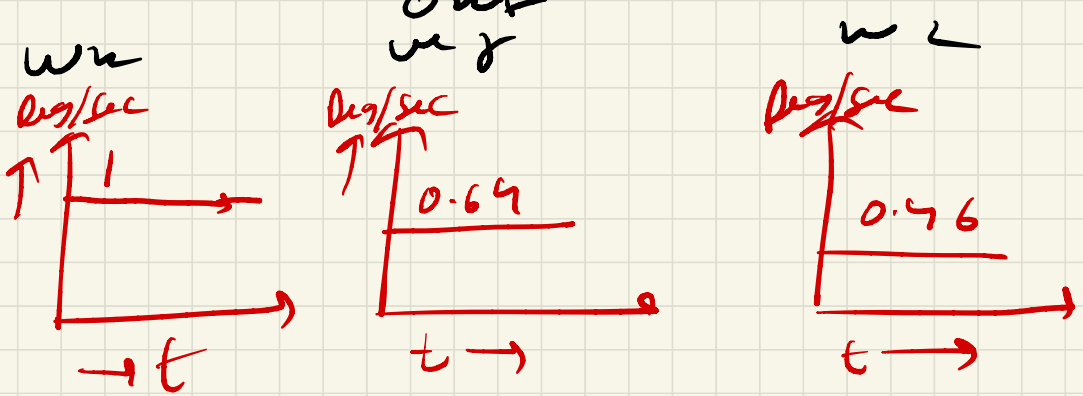
$= 31.82 \text{ sec}$

$w_x = 1$

$w_y = 0.5030 \times 1.2743 = 0.64$

$w_z = 0.3621 \times 1.2743 = 0.46$

∴ Profile of w_z w_y w_z are straight lines // to x axis



Let

$$\hat{n} = \begin{bmatrix} k_x \\ k_y \\ k_z \end{bmatrix}$$

Skew symmetric

$$[\hat{N}] = \begin{bmatrix} 0 & -k_z & k_y \\ k_z & 0 & -k_x \\ -k_y & k_x & 0 \end{bmatrix}$$

$$R(\hat{n}, \theta) = I + \sin\theta[\hat{n}] + (1-\cos\theta)[\hat{n}]^2$$

This was used to compute angle axis representation in the code.

$R(\hat{n}, \gamma) =$ 0 1 2

$v_\theta = (1 - \cos\theta)$ $s_\theta = \sin\theta$

$$R_{k,\theta} = \begin{bmatrix} k_x^2 v_\theta + c_\theta & k_x k_y v_\theta - k_z s_\theta & k_x k_z v_\theta + k_y s_\theta \\ k_x k_y v_\theta + k_z s_\theta & k_y^2 v_\theta + c_\theta & k_y k_z v_\theta - k_x s_\theta \\ k_x k_z v_\theta - k_y s_\theta & k_y k_z v_\theta + k_x s_\theta & k_z^2 v_\theta + c_\theta \end{bmatrix} \begin{matrix} 0 \\ 1 \\ 2 \end{matrix}$$

0 1 2

$$\begin{bmatrix} \cos(\phi)\cos(\theta) & -\sin(\phi)\cos(\psi) + \sin(\psi)\sin(\theta)\cos(\phi) & \sin(\phi)\sin(\psi) + \sin(\theta)\cos(\phi)\cos(\psi) \\ \sin(\phi)\cos(\theta) & \sin(\phi)\sin(\psi)\sin(\theta) + \cos(\phi)\cos(\psi) & \sin(\phi)\sin(\theta)\cos(\psi) - \sin(\psi)\cos(\phi) \\ -\sin(\theta) & \sin(\psi)\cos(\theta) & \cos(\psi)\cos(\theta) \end{bmatrix} \begin{matrix} 0 \\ 1 \\ 2 \end{matrix}$$

Comparing the two matrices we get θ, ϕ, ψ

$$\frac{R[2,1]}{R[2,2]} = \frac{\sin\psi\cos\theta}{\cos\psi\cos\theta} = \frac{k_y k_z v_\theta + k_x s_\theta}{k_z^2 v_\theta + c_\theta}$$

$$\tan\psi = \frac{k_y k_z v_\theta + k_x s_\theta}{k_z^2 v_\theta + c_\theta}$$

$$R[2,0] = -\sin\theta = k_x k_z v_\theta - k_y s_\theta$$

$$\frac{R[1,0]}{R[0,0]} = \frac{\sin\phi\cos\theta}{\cos\phi\cos\theta} = \frac{k_x k_y v_\theta + k_z s_\theta}{k_x^2 v_\theta + c_\theta}$$

$$\tan\phi = \frac{k_x k_y v_\theta + k_z s_\theta}{k_x^2 v_\theta + c_\theta}$$

Plots are shown in the code and file.

```
from sympy import *
from mpmath import radians
import math

x,y,z,d_x,d_y,d_z=symbols('x,y,z,d_x,d_y,d_z')
phi,theta,psi=symbols('phi,theta, psi')
w_x,w_y,w_z=symbols('w_x,w_y,w_z')
gammadot=symbols('gammadot')

"""
R_phi,T_y,R_theta,R_psi=symbols('R_phi,T_y,R_theta,R_psi')
"""

R_psi=Matrix([[1,0,0],
     [0,cos(psi),-sin(psi)],
     [0,sin(psi),cos(psi)]])

R_theta=Matrix(
    [[cos(theta),0,sin(theta)],
     [0,1,0],
     [-sin(theta),0,cos(theta)]])
R_phi=Matrix(
    [[cos(phi),-sin(phi),0],
     [sin(phi),cos(phi),0],
     [0,0,1]])

#final orientation of the frame.
final_orientation=simplify(R_phi*R_theta*R_psi)


# calcuating trace to calculate the angle for angle axis rotstion
trace=final_orientation[0,0]+final_orientation[1,1]+final_orientation[2,2]
trace=trace.subs({psi:radians(35),theta:radians(15),phi:radians(20)})

# Calculating the angle alpha for angle of angle axis
alpha=math.acos((trace-1)/2)


# calulation for the axis of rotation for psi,theta,phi
val=1/(2*math.sin(alpha))
Axis=Matrix([
        [final_orientation[2,1]-final_orientation[1,2]],
        [final_orientation[0,2]-final_orientation[2,0]],
        [final_orientation[1,0]-final_orientation[0,1]]])

k=val*Axis


k=k.subs({psi:radians(35),theta:radians(15),phi:radians(20)})







# rotation matrix for global frame about the n axis
w_g=k*gammadot

#skew symetric matrix for the global frame
w_g_skew=Matrix([[0,-w_g[2],w_g[1]],[w_g[2],0,-w_g[0]],[-w_g[1],w_g[0],0]])


#Rotation skew symetric matrix for drone local frame
w_drone=Matrix([[0,-w_z,w_y],[w_z,0,-w_x],[-w_y,w_x,0]])
w_drone_1=(final_orientation.T)*w_g_skew*(final_orientation)

# finding thr values for w_x,w_y,w_z for given angles
solution=w_drone-w_drone_1
solution=solution.subs({psi:radians(35),theta:radians(15),phi:radians(20)})

print('\n')
w_z=(solve(solution[0,1],w_z))
w_y=(solve(solution[0,2],w_y))
w_x=(solve(solution[1,2],w_x))
```

```
#we know that max of w is w=1deg/sec
#therefore the max angular velocity is of w_x=1 deg/sec
# we find the gammadot and hence can find the shortest time for rotation
gammadot=solve(w_x[0]-1,gammadot)


time_1=40.5/gammadot[0]
pprint(time_1)
```