In [1]:

```python
import sympy as sp
import sympy.physics.vector as vec
from mecvoo import *
```

# 1

## i

In [2]:

```python
lamda, phi = sp.symbols("lambda, phi")
NED = vec.ReferenceFrame("NED")
ECEF = NED.orientnew("ECEF", "Body", [phi + rad(90), -lamda, 0], "yzx")
D_NED_ECEF = NED.dcm(ECEF)
math(sp.latex(D_NED_ECEF))
D_NED_ECEF = sp.rot_axis2(-phi) @ sp.rot_axis1(lamda) @ sp.rot_axis2(-sp.pi / 2)
math(sp.latex(D_NED_ECEF))
```

$$\begin{bmatrix} -\sin(\phi)\cos(\lambda) & -\sin(\lambda)\sin(\phi) & \cos(\phi) \\ -\sin(\lambda) & \cos(\lambda) & 0 \\ -\cos(\lambda)\cos(\phi) & -\sin(\lambda)\cos(\phi) & -\sin(\phi) \end{bmatrix}$$

$$\begin{bmatrix} -\sin(\phi)\cos(\lambda) & -\sin(\lambda)\sin(\phi) & \cos(\phi) \\ -\sin(\lambda) & \cos(\lambda) & 0 \\ -\cos(\lambda)\cos(\phi) & -\sin(\lambda)\cos(\phi) & -\sin(\phi) \end{bmatrix}$$

## ii

In [3]:

```python
# Numa Terra esférica, a coordenada será igual a um vetor na direção ECEF.x de
# magnitude R + altitude, rotacionado em ECEF.y pela latitude e ECEF.z pela longitude
# então, as coordenadas ECEF aplicam a rotação contrária
def coord2vec(lon, lat, h=0, R=sp.Symbol("R")) -> vec.Vector:
    """
    Calcula vetor de posição a partir de coordenadas geográficas e altitude e retorna
    um vetor com base ECEF
    ==========
    Inputs
    lon: longitude em graus
    lat: latitude em graus
    h: altitude em metros
    Output
    """
    lon = rad(lon)
    lat = rad(lat)
    rot = sp.rot_axis3(-lon) @ sp.rot_axis2(lat)  # rotação "body fixed"
    mag = R + h
    base = sp.Matrix([1, 0, 0])
    return vector(mag * rot @ base, ECEF)
```

In [4]:

```python
def vec2coord(vector: vec.Vector) -> sp.Matrix:
    """
    Calcula longitude, latitude e altitude partir do vetor da posição relativa ao
    centro da Terra e retorna uma matriz nessa ordem
    ==========
    Inputs
    vector: vetor de posição relativo ao centro da Terra (em sistemas conectados a ECEF)
    """
    R = sp.symbols("R")
    c_ecef = vector.to_matrix(ECEF)

    # Altitude = distância ao centro - raio da Terra
    h = sp.simplify(vector.magnitude() - R)
    # Longitude pode ser encontrada pela razão das coordenadas X e Y
    lon = sp.simplify(sp.atan2(c_ecef[1], c_ecef[0]))
    # Latitude pode ser encontrada pela razão entre a coordenada Z e a projeção no plano XY
    r_ = sp.sqrt(c_ecef[0] ** 2 + c_ecef[1] ** 2)
    lat = sp.simplify(sp.atan(c_ecef[2] / r_))
    return sp.Matrix([lon, lat, h])
```

In [5]:

```python
coord_ECEF = vector(sp.symbols("X:Z"), ECEF)
lon, lat, h = sp.Matrix(sp.symbols("lon, lat, h"))
R = sp.symbols("R")

math(
    f"{sp.latex(coord_ECEF.to_matrix(ECEF))}_{{ECEF}} =  "
    + r"R_z(-lon) R_y(lat) \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} (R + h)"
)


P = coord2vec(lon, lat, h)
math(
    f"{sp.latex(coord_ECEF.to_matrix(ECEF))}_{{ECEF}} =  " + sp.latex(P.to_matrix(ECEF))
)

math(
    r"\begin{bmatrix} lon \\ lat \\ alt \end{bmatrix} = "
    + sp.latex(sp.Matrix(vec2coord(coord_ECEF)))
)
```

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}_{ECEF} = R_z(-lon) R_y(lat) \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} (R + h)$$

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}_{ECEF} = \begin{bmatrix} (R+h)\cos\left(\frac{\pi lat}{180}\right)\cos\left(\frac{\pi lon}{180}\right) \\ (R+h)\sin\left(\frac{\pi lon}{180}\right)\cos\left(\frac{\pi lat}{180}\right) \\ (R+h)\sin\left(\frac{\pi lat}{180}\right) \end{bmatrix}$$

$$\begin{bmatrix} lon \\ lat \\ alt \end{bmatrix} = \begin{bmatrix} \operatorname{atan}_2(Y, X) \\ \operatorname{atan}\left(\frac{Z}{\sqrt{X^2+Y^2}}\right) \\ -R + \sqrt{X^2 + Y^2 + Z^2} \end{bmatrix}$$

# iii

In [6]:

```python
subs2_3 = {
    lon: -48.0448584,
    lat: -15.9890146,
    h: 1221,
    R: 6378164,
}

math(sp.latex(P.evalf(7, subs=subs2_3).to_matrix(ECEF)))
```

$$\begin{bmatrix} 4099938.0 \\ -4560618.0 \\ -1757221.0 \end{bmatrix}$$

# iv

In [7]:

```python
P_GPS = vector((10981457, -13087191, -20360055), ECEF)

R_NED = (P_GPS - P).to_matrix(NED)

# Posição em NED
math(sp.latex(R_NED.subs({phi: rad(lat), lamda: rad(lon)}).evalf(8, subs=subs2_3)))
```

$$\begin{bmatrix} -14869285.0 \\ -582859.46 \\ -15642501.0 \end{bmatrix}$$

In [8]:

```
# azimute e elevação no horizonte local
azimute = (
    sp.atan2(R_NED[1], R_NED[0])
    .subs({phi: rad(lat), lamda: rad(lon)})
    .evalf(8, subs=subs2_3)
)
elevacao = (
    sp.atan(-R_NED[2] / sp.sqrt(R_NED[0] ** 2 + R_NED[1] ** 2))
    .subs({phi: rad(lat), lamda: rad(lon)})
    .evalf(8, subs=subs2_3)
)
math(
    f"\\alpha = {deg(azimute).evalf()}\\deg \\\\"
    f"\\lambda = {deg(elevacao).evalf()}\\deg"
)
```

$$\alpha = -177.755218410765 \deg$$
$$\lambda = 46.4296882530791 \deg$$

## v

In [9]:

```
ENU = NED.orientnew("ENU", "DCM", sp.Matrix([[0, 1, 0], [1, 0, 0], [0, 0, -1]]))
math(sp.latex(ENU.dcm(ECEF)))
math(sp.latex(ECEF.dcm(ENU)))
```

$$\begin{bmatrix} -\sin(\lambda) & \cos(\lambda) & 0 \\ -\sin(\phi)\cos(\lambda) & -\sin(\lambda)\sin(\phi) & \cos(\phi) \\ \cos(\lambda)\cos(\phi) & \sin(\lambda)\cos(\phi) & \sin(\phi) \end{bmatrix}$$

$$\begin{bmatrix} -\sin(\lambda) & -\sin(\phi)\cos(\lambda) & \cos(\lambda)\cos(\phi) \\ \cos(\lambda) & -\sin(\lambda)\sin(\phi) & \sin(\lambda)\cos(\phi) \\ 0 & \cos(\phi) & \sin(\phi) \end{bmatrix}$$

## vi

In [10]:

```
omega_e = vector([0, 0, 1], ECEF) * sp.symbols("omega_e")
math(sp.latex(omega_e.to_matrix(ENU)))
```

$$\begin{bmatrix} 0 \\ \omega_e \cos(\phi) \\ \omega_e \sin(\phi) \end{bmatrix}$$

## vii

In [11]:

```
g = vector([0, 0, 1], NED) * sp.symbols("g")
math(sp.latex(g.to_matrix(ECEF)))
```

$$\begin{bmatrix} -g\cos(\lambda)\cos(\phi) \\ -g\sin(\lambda)\cos(\phi) \\ -g\sin(\phi) \end{bmatrix}$$

## viii

In [12]:

```
math(sp.latex(NED.dcm(ENU)))
math(sp.latex(ENU.dcm(NED)))
```

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

# 2

## i

In [13]:

```
t = sp.symbols("t")
# vec.init_vprinting()


def d(x):
    return sp.diff(x, t)


U, V, W, P, Q, R = vec.dynamicsymbols("U, V, W, P, Q, R")
a, b, c = sp.symbols("a, b, c")
psi, theta, phi = sp.symbols("psi, theta, phi")

# Vetores da questão representados em Sb
Vb = sp.Matrix([U, V, W])
wb = sp.Matrix([P, Q, R])
r_ = sp.Matrix([a, b, c])
g = (
    sp.rot_axis1(phi)
    @ sp.rot_axis2(theta)
    @ sp.rot_axis3(psi)
    @ sp.Matrix([0, 0, sp.symbols("g_0")])
)
```

In [14]:

```
# Primeira derivada
di_r = Vb
di_r_ = wb.cross(r_)
di_rAC = di_r + di_r_

# Segunda derivada
ddi_rAC = d(di_rAC) + wb.cross(di_rAC)
math(vec.vlatex(ddi_rAC - g))
```

$$\left[\begin{matrix}- b \dot{R} + c \dot{Q} + g_{0} \sin{\left(\theta \right)} + \left(- a Q + b P + W\right) Q - \left(a R - c P + V\right) R + \dot{U}\\a \dot{R} - c \dot{P} - g_{0} \sin{\left(\phi \right)} \cos{\left(\theta \right)} - \left(- a Q + b P + W\right) P + \left(- b R + c Q + U\right) R + \dot{V}\\- a \dot{Q} + b \dot{P} - g_{0} \cos{\left(\phi \right)} \cos{\left(\theta \right)} + \left(a R - c P + V\right) P - \left(- b R + c Q + U\right) Q + \dot{W}\end{matrix}\right]$$

# 3

## i

In [15]:

```
theta, phi, nu, psi = vec.dynamicsymbols("theta, phi, nu, psi")
ine = vec.ReferenceFrame("ine")
cil = ine.orientnew("cil", "Axis", (theta, ine.z))
i, j, k = cil.x, cil.y, cil.z

m = sp.sin(phi) * i + sp.cos(phi) * k
p = m.cross(j)
n = sp.cos(nu) * j + sp.sin(nu) * p

subs = {
    d(d(theta)): 0,
    d(d(phi)): 0,
    d(d(nu)): 0,
    d(d(psi)): 0,
}

w = d(theta) * k + d(phi) * j + d(nu) * m + d(psi) * n
math(vec.vlatex(w.to_matrix(cil)))
```

$$\left[\begin{matrix}- \sin{\left(\nu \right)} \cos{\left(\phi \right)} \dot{\psi} + \sin{\left(\phi \right)} \dot{\nu}\\\cos{\left(\nu \right)} \dot{\psi} + \dot{\phi}\\\sin{\left(\nu \right)} \sin{\left(\phi \right)} \dot{\psi} + \cos{\left(\phi \right)} \dot{\nu} + \dot{\theta}\end{matrix}\right]$$

## ii

In [16]:
```
math(vec.vlatex(sp.expand(w.dt(ine).to_matrix(cil).subs(subs))))
```

$$\left[\begin{matrix}\sin{\left(\nu \right)} \sin{\left(\phi \right)} \dot{\phi} \dot{\psi} - \cos{\left(\nu \right)} \cos{\left(\phi \right)} \dot{\nu} \dot{\psi} - \cos{\left(\nu \right)} \dot{\psi} \dot{\theta} + \cos{\left(\phi \right)} \dot{\nu} \dot{\phi} - \dot{\phi} \dot{\theta}\\- \sin{\left(\nu \right)} \cos{\left(\phi \right)} \dot{\psi} \dot{\theta} - \sin{\left(\nu \right)} \dot{\nu} \dot{\psi} + \sin{\left(\phi \right)} \dot{\nu} \dot{\theta}\\\sin{\left(\nu \right)} \cos{\left(\phi \right)} \dot{\phi} \dot{\psi} + \sin{\left(\phi \right)} \cos{\left(\nu \right)} \dot{\nu} \dot{\psi} - \sin{\left(\phi \right)} \dot{\nu} \dot{\phi}\end{matrix}\right]$$

# 4

## i

Diedro positivo aumenta a estabilidade relativa a rolagem, diedro negativo diminui a estabilidade relativa a rolagem. Esse aumento e diminuição de estabilidade implicam, respectivamente, em diminuição e aumento da manobrabilidade da aeronave relativa a rolagem.

A escolha do diedro então deve levar em consideração os outros fatores que afetam a estabilidade de rolagem e o equilíbrio desejado entre estabilidade/manobrabilidade da aeronave.

## ii

Enflechamento positivo aumenta a estabilidade relativa a guinada, enflechamento negativo diminui a estabilidade relativa a guinada. Esse aumento e diminuição de estabilidade implicam, respectivamente, em diminuição e aumento da manobrabilidade da aeronave relativa a guinada. Além disso, o enflechamento pode causar momento adverso de rolagem quando em manobra de rolagem.

A escolha do enflechamento então deve levar em consideração os outros fatores que afetam a estabilidade de guinada e o equilíbrio desejado entre estabilidade/manobrabilidade da aeronave.

## iii

Maior alongamento implica em melhor performance de sustentação da aeronave, diminuindo perdas de ponta de asa. Entretanto, alongamento maior também causa maiores momentos fletores na asa (devido a braço de alavanca maior), e menor manobrabilidade (devido a maior momento de inércia).

A escolha do alongamento, então, deve levar em conta as velocidades de operação do projeto, a performance desejada, e o equilíbrio desejado entre estabilidade e manobrabilidade.

# 5

É provável que a colocação dos equipamentos, no corpo level da aeronave, tenha movido o CG da aeronave além do seu envelope de estabilidade estática. Como a instabilidade começa com pequenas ativações do profundor, é provável que o coeficiente de momento de arfagem da aeronave tenha se tornado positivo sob condições de operação. Assim, quando um pequeno momento de arfagem pica a aeronave, a mudança de ângulo de ataque aumenta o momento de arfagem, que consequentemente pica a aeronave ainda mais, até a aeronave entrar em stall. Picar a aeronave causa efeito parecido no sentido oposto.

Uma possível solução para esse problema seria mover o centro de massa da aeronave mais à frente, com o objetivo de deixá-lo à frente do centro aerodinâmico. Exagerar essa correção pode causar instabilidade dinâmica (apesar da estabilidade estática), onde o momento inverso gerado pela variação de ângulo de ataque faria a aeronave corrigir para um ângulo de sinal oposto, mas magnitude maior.