```
"""
Functions representing the formulas in the article
"""

from math import pi, sin, cos, acos, sqrt

# Constants
G = 9.81

# Cone functions
def trun_cone_vol(r : float, R : float, h : float) -> float:
    """Returns the volume of a truncated cone

    Args:
        r (float): Upper radius of the cone
        R (float): Lower radius of the cone
        h (float): Height of the cone

    Returns:
        float: The volume
    """
    return h * pi / 3 * (r**2 + R*r + R**2)


def trun_cone_com(r : float, R : float, h : float) -> float:
    """Returns the center of mass of a truncated cone

    Args:
        r (float): _description_
        R (float): _description_
        h (float): _description_

    Returns:
        float: _description_
    """
    return h * (R**2/2 + R*(r-R)*2/3 + (r-R)**2/4) / (R**2 + R*(r-R) +
(r-R)**2/3)


# Geometrical shape
def alpha_t(beta : float) -> float:
    return pi/2 - beta


def beta_t(alpha : float) -> float:
    return pi/2 - alpha


def gamma_t(alpha : float) -> float:
    return (pi - alpha) / 2


def cyl_angle_from_length(h_com : float, length : float) -> float:
    return acos(1 - 2*length**2 / h_com**2)


def cyl_length_from_angle(h_com : float, alpha : float) -> float:
    return h_com * sqrt((1 - cos(alpha))/2)
```

```
# Motor torque functinos
def inertia_tree(h : float, m : float) -> float:
    return h**2 * m / 3


def inertia_rod(i_t : float, h_com : float) -> float:
    return 4 * i_t / h_com**2


def inertia_motor(i_r : float, spec_pitch : float) -> float:
    return i_r * spec_pitch**2


def tree_load_torque(h_com : float, m : float, beta : float) -> float:
    return h_com * m * G * cos(beta)


def cylinder_load_force(t_tree : float, h_com : float, gamma : float) ->
float:
    return 2 * t_tree / h_com / sin(gamma)


def motor_load_force(f_cyl : float, pitch : float) -> float:
    return f_cyl * pitch

# Motor torque curves
MOTOR_MAX_TORQUE = 71.1
MOTOR_MAX_SPEED = 3000 / 60 * 2 * pi
MOTOR_AMPS_PER_NM = 0.934
MOTOR_VOLTAGE = 200

def motor_torque(omega : float) -> float:
    return MOTOR_MAX_TORQUE * (1 - omega / MOTOR_MAX_SPEED)

def motor_amperes(torque : float) -> float:
    return MOTOR_AMPS_PER_NM * torque
```