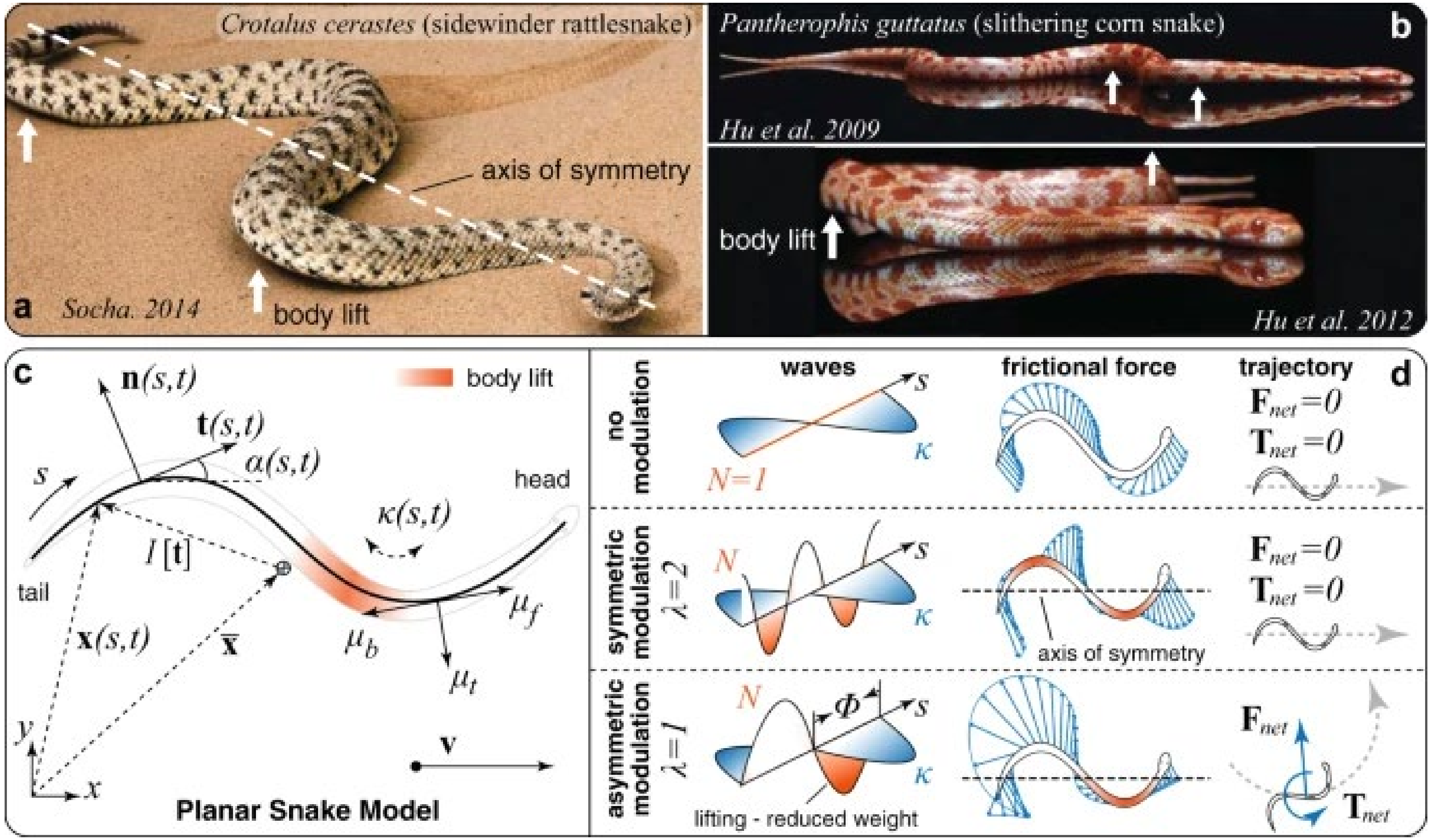
Assignment: Slithering Snake Motion

Spring 2022

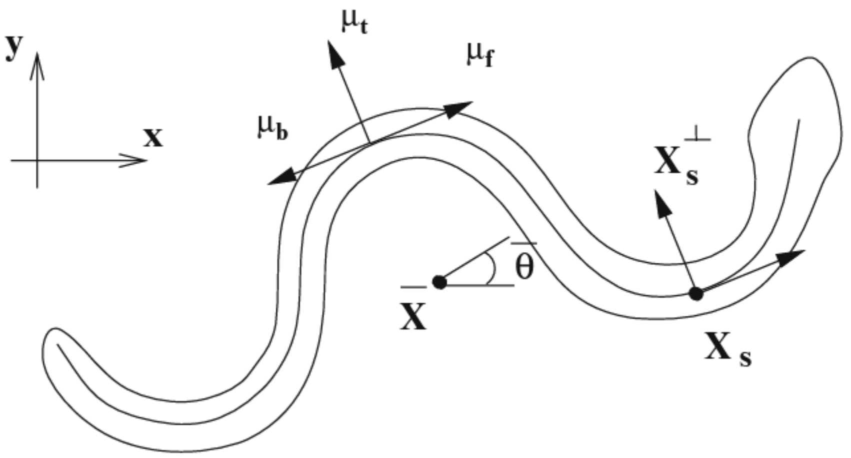
Instructors: Saravana Prashanth Mural Babu

Theory of Slithering Locomotion

Slithering Snake Motion



Slithering Snake Motion



$$Fr = \frac{L}{\mu_f g \tau^2} = \frac{\text{inertia}}{\text{friction}} \sim 10^{-3}$$
$$An_\parallel = \frac{\mu_b}{\mu_f} = \frac{\text{backward friction}}{\text{forward friction}} \sim 1.3$$
$$An_\perp = \frac{\mu_t}{\mu_f} = \frac{\text{transverse friction}}{\text{forward friction}} \sim 1.7.$$

$\kappa(s, t) = \alpha \cos k\pi(s + t)$ (in dimensionless units).

stereotypical snake curvature function
$7cos(2\pi s)$

Step 1:

Import the necessary classes. Timoshenko bean, wrapper functions, rod class, forces, timestepping functions, and callback classes.

```
import numpy as np

# import wrappers
from elastica.wrappers import BaseSystemCollection, Constraints, Forcing, CallBa

# import rod class and forces to be applied
from elastica.rod.cosserat_rod import CosseratRod
from elastica.external_forces import GravityForces, MuscleTorques
from elastica.interaction import AnisotropicFrictionalPlane

# import timestepping functions
from elastica.timestepper.symplectic_steppers import PositionVerlet
from elastica.timestepper import integrate

# import call back functions
from elastica.callback_functions import CallBackBaseClass
from collections import defaultdict
```

```
import numpy
import matplotlib
import elastica
```

Step 2: Initialize System and Add Rod

```
class SnakeSimulator(BaseSystemCollection, Constraints, Forcing, CallBacks):
    pass


snake_sim = SnakeSimulator()

# Define rod parameters
n_elem = 50
start = np.array([0.0, 0.0, 0.0])
direction = np.array([0.0, 0.0, 1.0])
normal = np.array([0.0, 1.0, 0.0])
base_length = 0.35  ## 0.7
base_radius = base_length * 0.011
base_area = np.pi * base_radius ** 2
density = 1000
nu = 1e-4
E = 1e6
poisson_ratio = 0.5
shear_modulus = E / (poisson_ratio + 1.0)

# Create rod
shearable_rod = CosseratRod.straight_rod(n_elem, start, direction,
                 normal, base_length, base_radius, density, nu, E,
                 shear_modulus=shear_modulus,)
# Add rod to the snake system
snake_sim.append(shearable_rod)
```

Step 3:
Add Forces to Rod

Gravitational forces

```python
# Add gravitational forces
gravitational_acc = -9.80665
snake_sim.add_forcing_to(shearable_rod).using(
    GravityForces, acc_gravity=np.array([0.0, gravitational_acc, 0.0])
)
print("Gravity now acting on shearable rod")
```

Muscle Torques

```python
# Define muscle torque parameters
period =2.0
wave_length = 1.0  ##1.0
b_coeff = np.array([3.4e-3, 3.3e-3, 4.2e-3, 2.6e-3, 3.6e-3, 3.5e-3])

# Add muscle torques to the rod
snake_sim.add_forcing_to(shearable_rod).using(
    MuscleTorques,
    base_length=base_length,
    b_coeff=b_coeff,
    period=period,
    wave_number=2.0 * np.pi / (wave_length),
    phase_shift=0.0,
    rest_lengths=shearable_rod.rest_lengths,
    ramp_up_time=period,
    direction=normal,
    with_spline=True,
)
print("Muscle torques added to the rod")
```

Friction Forces

```python
# Define friction force parameters
origin_plane = np.array([0.0, -base_radius, 0.0])
normal_plane = normal
slip_velocity_tol = 1e-8
froude = 0.1
mu = base_length / (period * period * np.abs(gravitational_acc) * froude)
kinetic_mu_array = np.array(
    [1.0 * mu, 1.5 * mu, 2.0 * mu]
)  # [forward, backward, sideways]
static_mu_array = 2 * kinetic_mu_array

# Add friction forces to the rod
snake_sim.add_forcing_to(shearable_rod).using(
    AnisotropicFrictionalPlane,
    k=1.0,
    nu=1e-6,  ##### friction
    plane_origin=origin_plane,
    plane_normal=normal_plane,
    slip_velocity_tol=slip_velocity_tol,
    static_mu_array=static_mu_array,
    kinetic_mu_array=kinetic_mu_array,
)
print("Friction forces added to the rod")
```

Step 4:
Add Callback Function

```python
# Add call backs
class ContinuumSnakeCallBack(CallBackBaseClass):
    """
    Call back function for continuum snake
    """

    def __init__(self, step_skip: int, callback_params: dict):
        CallBackBaseClass.__init__(self)
        self.every = step_skip
        self.callback_params = callback_params

    def make_callback(self, system, time, current_step: int):

        if current_step % self.every == 0:

            self.callback_params["time"].append(time)
            self.callback_params["step"].append(current_step)
            self.callback_params["position"].append(system.position_collection.copy())
            self.callback_params["velocity"].append(system.velocity_collection.copy())
            self.callback_params["avg_velocity"].append(
                system.compute_velocity_center_of_mass()
            )

            self.callback_params["center_of_mass"].append(
                system.compute_position_center_of_mass()
            )
            self.callback_params["curvature"].append(system.kappa.copy())

            return


pp_list = defaultdict(list)
snake_sim.collect_diagnostics(shearable_rod).using(
    ContinuumSnakeCallBack, step_skip=1000, callback_params=pp_list
)
print("Callback function added to the simulator")
```

Step 5: Finalize the system and define time stepping

```python
snake_sim.finalize()

final_time = 10.0 * period
dt = 8.0e-6
total_steps = int(final_time / dt)
print("Total steps", total_steps)

timestepper = PositionVerlet()
```

```python
integrate(timestepper, snake_sim, final_time, total_steps)
```

```
100%|██████████| 2500000/2500000 [10:03<00:00, 4142.38it/s]
Final time of simulation is :  20.00000000137994

20.00000000137994
```

Step 6: Post-Process Data

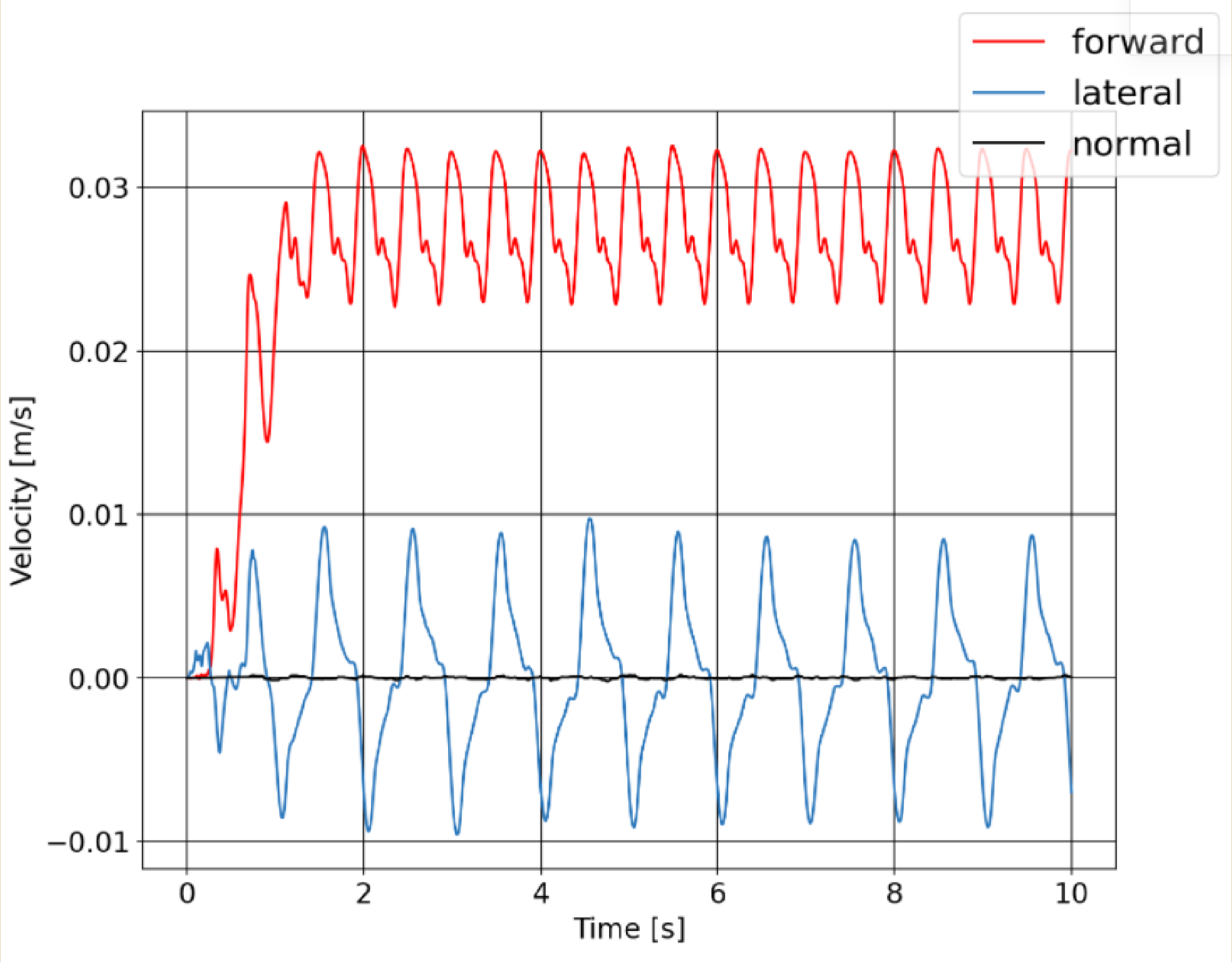
Compute projected velocity

```
def compute_projected_velocity(plot_params: dict, period):
    import numpy as np

    time_per_period = np.array(plot_params["time"]) / period
    avg_velocity = np.array(plot_params["avg_velocity"])
    center_of_mass = np.array(plot_params["center_of_mass"])

    # Compute rod velocity in rod direction. We need to compute that because,
    # after snake starts to move it chooses an arbitrary direction, which does not
    # have to be initial tangent direction of the rod. Thus we need to project the
    # snake velocity with respect to its new tangent and roll direction, after that
    # we will get the correct forward and lateral speed. After this projection
    # lateral velocity of the snake has to be oscillating between + and - values with
    # zero mean.
```

```
def compute_and_plot_velocity(plot_params: dict, period):
    from matplotlib import pyplot as plt
    from matplotlib.colors import to_rgb

    time_per_period = np.array(plot_params["time"]) / period
    avg_velocity = np.array(plot_params["avg_velocity"])
```

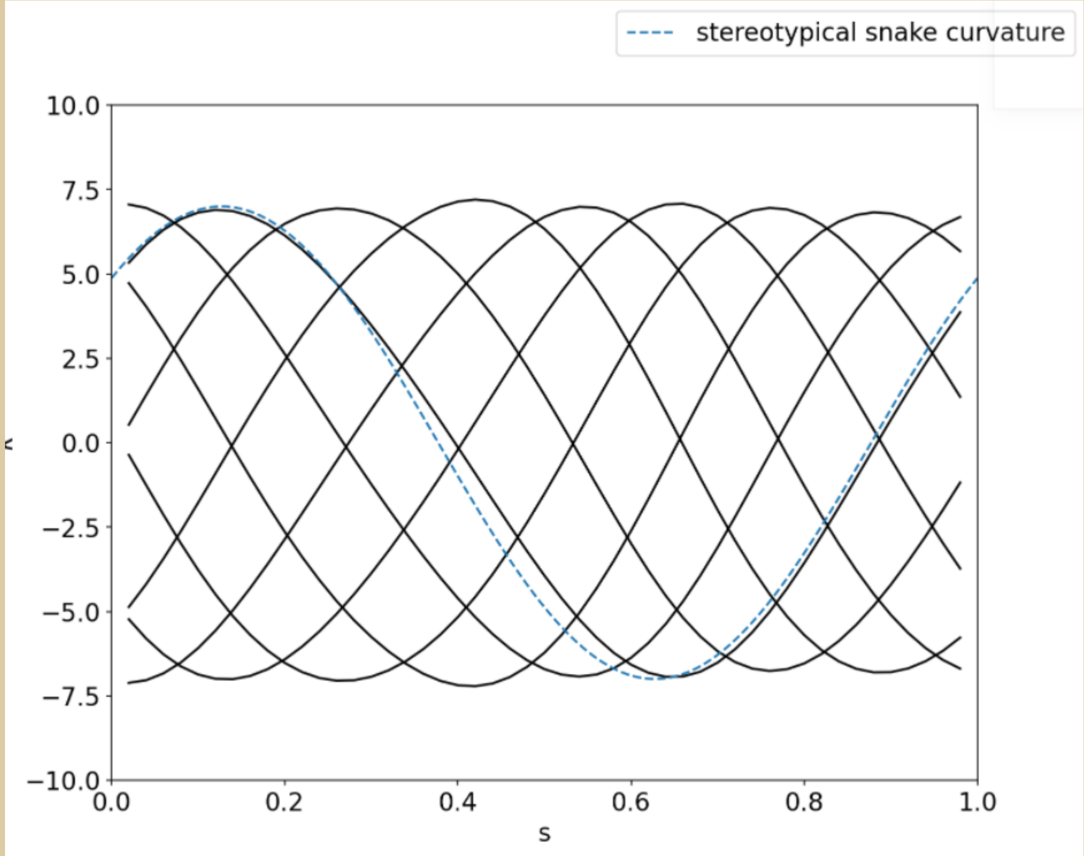
Projected velocity

```python
def compute_and_plot_velocity(plot_params: dict, period):
    from matplotlib import pyplot as plt
    from matplotlib.colors import to_rgb
```



Compute the curvature along the snake at different time instance

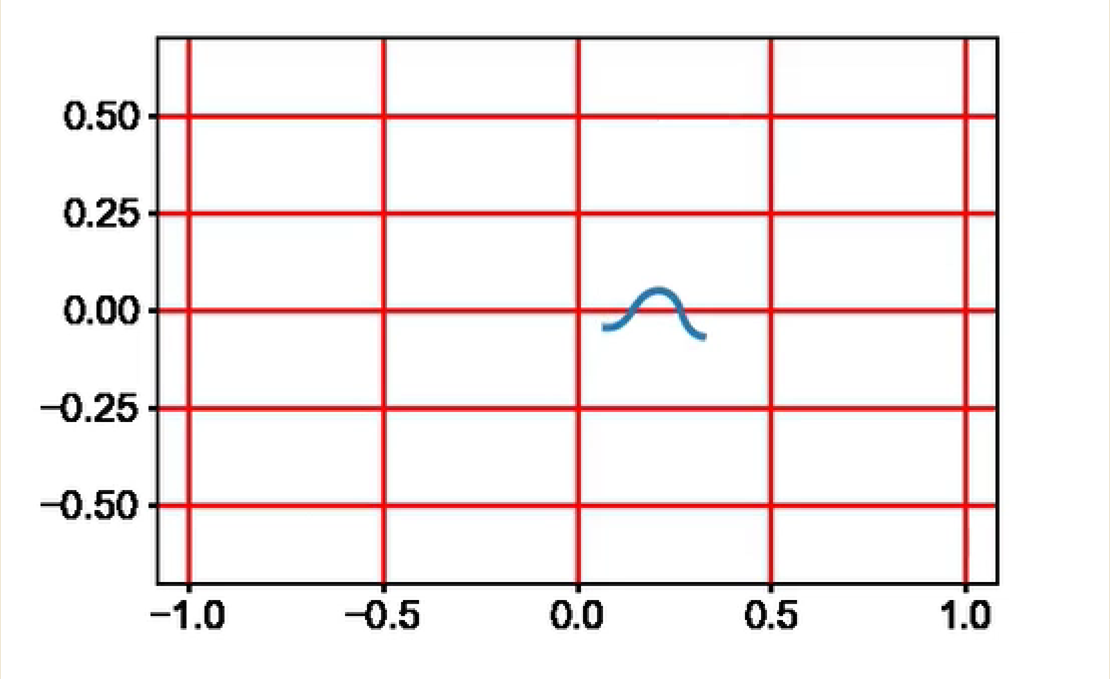
```python
def plot_curvature(
    plot_params: dict,
    rest_lengths,
    period,
):
    from matplotlib import pyplot as plt
    from matplotlib.colors import to_rgb

    s = np.cumsum(rest_lengths)
    L0 = s[-1]
    s = s / L0
    s = s[:-1].copy()
    x = np.linspace(0, 1, 100)
    curvature = np.array(plot_params["curvature"])
    time = np.array(plot_params["time"])
    peak_time = period * 0.125
    dt = time[1] - time[0]
    peak_idx = int(peak_time / (dt))
    plt.rcParams.update({"font.size": 16})
    fig = plt.figure(figsize=(10, 8), frameon=True, dpi=150)
    ax = fig.add_subplot(111)
    try:
        for i in range(peak_idx * 8, peak_idx * 8 * 2, peak_idx):
            ax.plot(s, curvature[i, 0, :] * L0, "k")
    except:
        print("Simulation time not long enough to plot curvature")
    ax.plot(
        x, 7 * np.cos(2 * np.pi * x - 0.80), "--", label="stereotypical snake curvature"
        #x, 7*x, "--", label="stereotypical snake curvature"
    )
    ax.set_ylabel(r"$\kappa$", fontsize=16)
    ax.set_xlabel("s", fontsize=16)
    ax.set_xlim(0, 1)
    ax.set_ylim(-10, 10)
    fig.legend(prop={"size": 16})
    plt.show()

    plt.close(plt.gcf())


plot_curvature(pp_list, shearable_rod.rest_lengths, period)
```




```python
def plot_curvature(
    plot_params: dict,
    rest_lengths,
    period,
):
    from matplotlib import pyplot as plt
    from matplotlib.colors import to_rgb
```

Step 7:
Make a video of snake gait

```
def plot_video_2D(plot_params: dict, video_name="video.mp4", margin=0.2, fps=15):
    from matplotlib import pyplot as plt
    import matplotlib.animation as manimation
```



Task 1: Changing Friction coheficients

```python
###############################################################################################################################
####################Task 1 ####################################################################################################
###############################################################################################################################

# Define friction force parameters
origin_plane = np.array([0.0, -base_radius, 0.0])
normal_plane = normal
slip_velocity_tol = 1e-8
froude = 0.1
mu = base_length / (period * period * np.abs(gravitational_acc) * froude)

kinetic_mu_array = np.array( [0.5 * mu, 0.5 * mu, 2 * mu])  # [forward, backward, sideways]    [1.0 * mu, 1.5 * mu, 2.0 * mu]


static_mu_array = 2 * kinetic_mu_array  # 2


# Add friction forces to the rod     Anisotropic Frictional Plane????
snake_sim.add_forcing_to(shearable_rod).using(
    AnisotropicFrictionalPlane,
    k=1.0,
    nu=1e-6,
    plane_origin=origin_plane,
    plane_normal=normal_plane,
    slip_velocity_tol=slip_velocity_tol,
    static_mu_array=static_mu_array,
    kinetic_mu_array=kinetic_mu_array,
)
print("Friction forces added to the rod")

###############################################################################################################################
################End of Task 1 #################################################################################################
###############################################################################################################################
```

Task 2: Changing magnitudes forces applied to the body

```
##########################################################################################################
####################Task 2 ################################################################################
##########################################################################################################


# Add gravitational forces
gravitational_acc = -9.80665
snake_sim.add_forcing_to(shearable_rod).using(
    GravityForces, acc_gravity=np.array([0.0, gravitational_acc, 0.0])
)
print("Gravity now acting on shearable rod")

# Define muscle torque parameters
period = 2.0
wave_length = 1.0
b_coeff = np.array([3.4e-3, 3.3e-3, 4.2e-3, 2.6e-3, 3.6e-3, 3.5e-3])   ####([3.4e-3, 3.3e-3, 4.2e-3, 2.6e-3, 3.6e-3, 3.5e-3])



# Add muscle torques to the rod
snake_sim.add_forcing_to(shearable_rod).using(
    MuscleTorques,
    base_length=base_length,
    b_coeff=b_coeff/2,   ##### /2
    period=period,
    wave_number=2.0 * np.pi / (wave_length),
    phase_shift=0.0,
    rest_lengths=shearable_rod.rest_lengths,
    ramp_up_time=period,
    direction=normal,
    with_spline=True,
)

print("Muscle torques added to the rod")

##########################################################################################################
##########################################################################################################
#############End of Task 2################################################################################
```

Task 3:
Changing rod parameters

```
################################################################################
#####################Task 3 ######################################################
################################################################################


# Define muscle torque parameters
period = 2.0
wave_length = 1.5
b_coeff = np.array([3.4e-3, 3.3e-3, 4.2e-3, 2.6e-3, 3.6e-3, 3.5e-3])

################################################################################
#####################End of Task 3 ###############################################
################################################################################
```

Task 4: Changing material characteristics and rod dimensions

```
###############################################################################################################
####################Task 4 ####################################################################################
###############################################################################################################
# Define rod parameters
n_elem = 20
start = np.array([0.0, 0.0, 0.0])
direction = np.array([0.0, 0.0, 1.0])
normal = np.array([0.0, 1.0, 0.0])
base_length = 0.35
base_radius = base_length * 0.011
base_area = np.pi * base_radius ** 2
density = 1000
nu = 1e-4
E = 0.2642e6   #Ecoflex 0050
poisson_ratio = 0.5
shear_modulus = E / (poisson_ratio + 1.0)

# Create rod
shearable_rod = CosseratRod.straight_rod(
    n_elem,
    start,
    direction,
    normal,
    base_length,
    base_radius,
    density,
    nu,
    E,
    shear_modulus=shear_modulus,
)

# Add rod to the snake system
snake_sim.append(shearable_rod)
###############################################################################################################
################End of Task 4 #################################################################################
###############################################################################################################
```