Team: Shock Waves

Members: Nilanjana Chakraborty 23BAI1073

C. Varshah 23BAI1371

PID controller code:

```
import gym

import pybullet as p

import numpy as np

import time

import csv

from gym_pybullet_drones.envs.HoverAviary import HoverAviary

env = HoverAviary(gui=True)

env.reset()

def generate_random_waypoints(num_waypoints, x_range, y_range, z_range):

   waypoints = []

   for _ in range(num_waypoints):

      x = np.random.uniform(*x_range)

      y = np.random.uniform(*y_range)

      z = np.random.uniform(*z_range)

      waypoints.append([x, y, z])

   return waypoints

num_waypoints = 5

x_range = (-2, 2)

y_range = (-2, 2)

z_range = (1, 3)

waypoints = generate_random_waypoints(num_waypoints, x_range, y_range, z_range)

with open('pid_values.csv', mode='w', newline='') as file:

   writer = csv.writer(file)

   writer.writerow(['Waypoint', 'Step', 'Error_X', 'Error_Y', 'Error_Z', 'Action_X', 'Action_Y', 'Action_Z', 
'Action_W'])

   def navigate_to_waypoints(env, waypoints, max_steps=500):
        for waypoint in waypoints:
            target_x, target_y, target_z = waypoint
            step_counter = 0
            print(f" Moving to waypoint: {waypoint}")

            while step_counter < max_steps:
                step_counter += 1
                action = np.zeros((1, 4))
                obs, reward, done, info, _ = env.step(action)
                print(f" Observation received: {obs} (Shape: {obs.shape})")
                if obs.size >= 3:
                    pos = np.array(obs[0, :3], dtype=float)
                else:
                    print(f"Invalid observation size: {obs.size}, assigning default position.")
                    pos = np.array([0, 0, 1.5])
                error = np.array([target_x - pos[0], target_y - pos[1], target_z - pos[2], 0])
                random_scale = np.random.uniform(0.1, 0.3)
                base_action = np.clip(error * random_scale + 0.5, 0.4, 1.6)
                noise = np.random.normal(0, 0.05, base_action.shape)  # Add some noise
                action = base_action + noise
                action = np.reshape(action, (1, 4))
                print(f" Action Shape: {action.shape} | Action: {action}")
                writer.writerow([waypoint, step_counter, error[0], error[1], error[2], action[0, 0], action[0, 1], action[0, 2], action[0, 3]])
                env.step(action)
                env.render()
                if np.linalg.norm(error[:3]) < 0.05:  # Only check position, not yaw
                    print(f" Waypoint {waypoint} reached!")
                    break
                time.sleep(0.1)
    navigate_to_waypoints(env, waypoints)
```

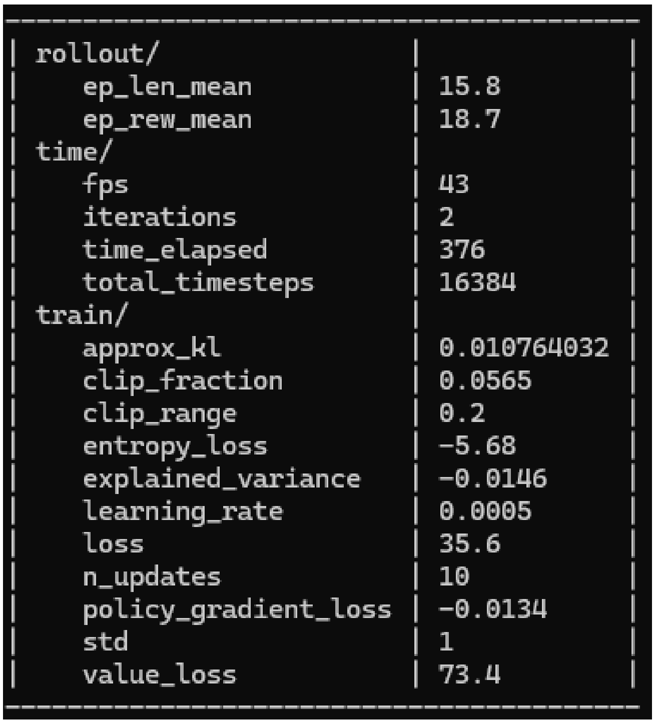
```
env.close()
```

RL model code:

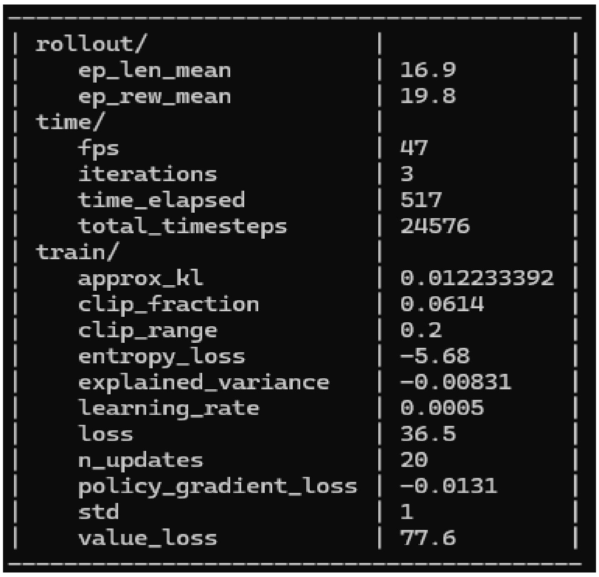
```
import torch
import numpy as np
from stable_baselines3 import PPO, DDPG, SAC
from gym_pybullet_drones.envs.HoverAviary import HoverAviary
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(f"Using device: {device}")
env = HoverAviary(gui=False)
env.reset()
TIMESTEPS = 200000  # Increase for better learning
MODEL_SAVE_PATH = "hoveraviary_model"
models = {
    "PPO": PPO(
        "MlpPolicy", env, verbose=1, device=device,
        n_steps=8192, batch_size=2048, learning_rate=0.0005, gamma=0.995,
        ent_coef=0.01, vf_coef=0.5, max_grad_norm=0.5
    ),
    "DDPG": DDPG(
        "MlpPolicy", env, verbose=1, device=device,
        learning_rate=0.001, batch_size=1024, gamma=0.99, tau=0.005
    ),
    "SAC": SAC(
        "MlpPolicy", env, verbose=1, device=device,
        learning_rate=0.0003, batch_size=512, gamma=0.98, tau=0.005, ent_coef="auto"
    ),
}
for algo, model in models.items():
    print(f" Training {algo} model...")
    model.learn(total_timesteps=TIMESTEPS)
    model.save(f"{MODEL_SAVE_PATH}_{algo}")
    print(f"{algo} model saved successfully!\n")
for algo in models.keys():
    loaded_model = models[algo].load(f"{MODEL_SAVE_PATH}_{algo}", env)
    print(f"{algo} model loaded successfully!")


print("All models trained, saved, and ready to use!")
```

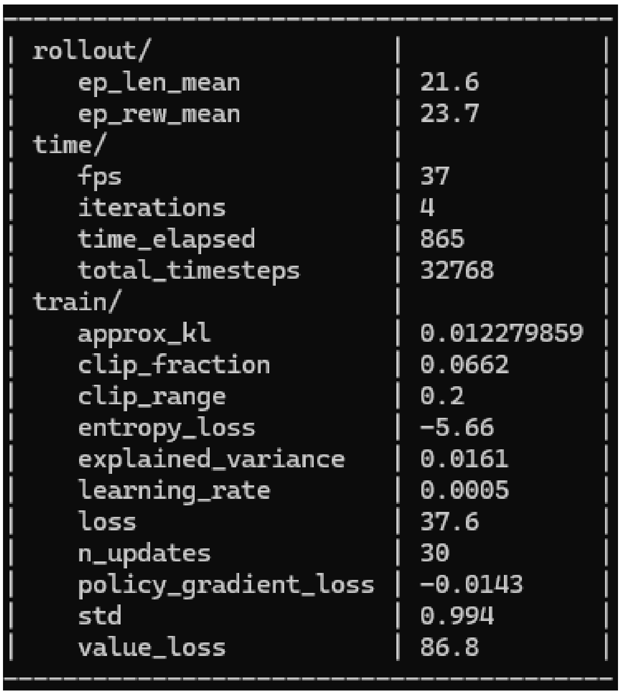
Training process screenshots:



```
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 15.8        |
|    ep_rew_mean          | 18.7        |
| time/                   |             |
|    fps                  | 43          |
|    iterations           | 2           |
|    time_elapsed         | 376         |
|    total_timesteps      | 16384       |
| train/                  |             |
|    approx_kl            | 0.010764032 |
|    clip_fraction        | 0.0565      |
|    clip_range           | 0.2         |
|    entropy_loss         | -5.68       |
|    explained_variance   | -0.0146     |
|    learning_rate        | 0.0005      |
|    loss                 | 35.6        |
|    n_updates            | 10          |
|    policy_gradient_loss | -0.0134     |
|    std                  | 1           |
|    value_loss           | 73.4        |
-----------------------------------------
```

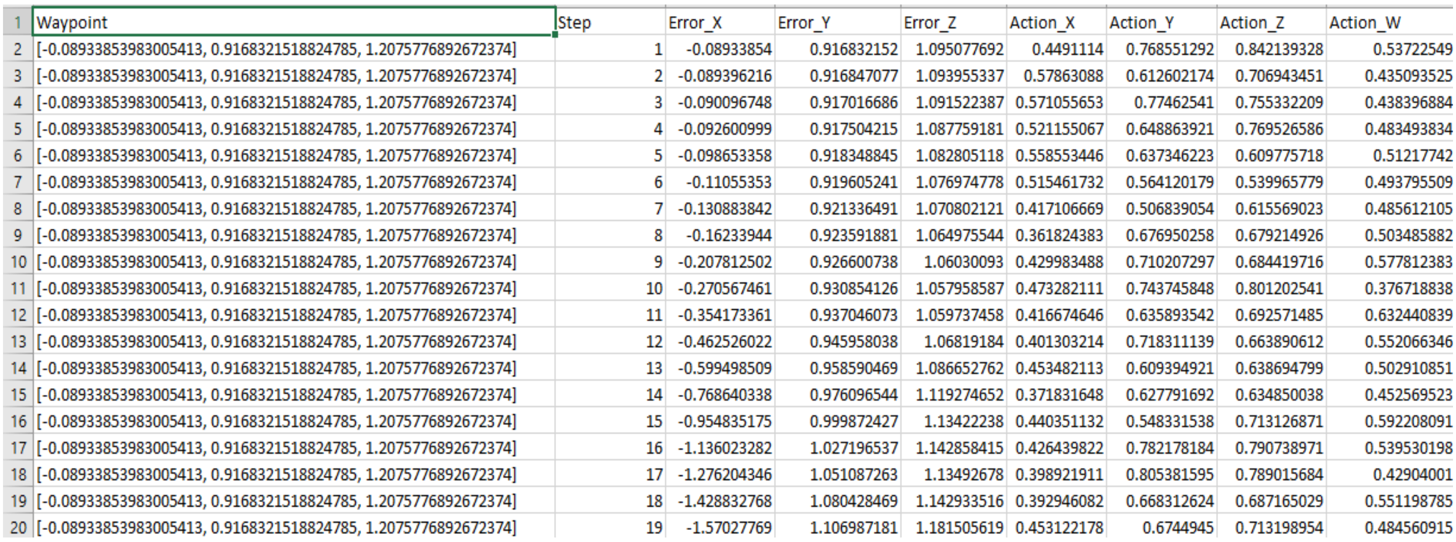


```
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 16.9        |
|    ep_rew_mean          | 19.8        |
| time/                   |             |
|    fps                  | 47          |
|    iterations           | 3           |
|    time_elapsed         | 517         |
|    total_timesteps      | 24576       |
| train/                  |             |
|    approx_kl            | 0.012233392 |
|    clip_fraction        | 0.0614      |
|    clip_range           | 0.2         |
|    entropy_loss         | -5.68       |
|    explained_variance   | -0.00831    |
|    learning_rate        | 0.0005      |
|    loss                 | 36.5        |
|    n_updates            | 20          |
|    policy_gradient_loss | -0.0131     |
|    std                  | 1           |
|    value_loss           | 77.6        |
-----------------------------------------
```



```
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 21.6        |
|    ep_rew_mean          | 23.7        |
| time/                   |             |
|    fps                  | 37          |
|    iterations           | 4           |
|    time_elapsed         | 865         |
|    total_timesteps      | 32768       |
| train/                  |             |
|    approx_kl            | 0.012279859 |
|    clip_fraction        | 0.0662      |
|    clip_range           | 0.2         |
|    entropy_loss         | -5.66       |
|    explained_variance   | 0.0161      |
|    learning_rate        | 0.0005      |
|    loss                 | 37.6        |
|    n_updates            | 30          |
|    policy_gradient_loss | -0.0143     |
|    std                  | 0.994       |
|    value_loss           | 86.8        |
-----------------------------------------
```

CSV File:



1	Waypoint	Step	Error_X	Error_Y	Error_Z	Action_X	Action_Y	Action_Z	Action_W
2	[-0.08933853983005413, 0.9168321518824785, 1.2075776892672374]	1	-0.08933854	0.916832152	1.095077692	0.4491114	0.768551292	0.842139328	0.53722549
3	[-0.08933853983005413, 0.9168321518824785, 1.2075776892672374]	2	-0.089396216	0.916847077	1.093955337	0.57863088	0.612602174	0.706943451	0.435093525
4	[-0.08933853983005413, 0.9168321518824785, 1.2075776892672374]	3	-0.090096748	0.917016686	1.091522387	0.571055653	0.77462541	0.755332209	0.438396884
5	[-0.08933853983005413, 0.9168321518824785, 1.2075776892672374]	4	-0.092600999	0.917504215	1.087759181	0.521155067	0.648863921	0.769526586	0.483493834
6	[-0.08933853983005413, 0.9168321518824785, 1.2075776892672374]	5	-0.098653358	0.918348845	1.082805118	0.558553446	0.637346223	0.609775718	0.51217742
7	[-0.08933853983005413, 0.9168321518824785, 1.2075776892672374]	6	-0.11055353	0.919605241	1.076974778	0.515461732	0.564120179	0.539965779	0.493795509
8	[-0.08933853983005413, 0.9168321518824785, 1.2075776892672374]	7	-0.130883842	0.921336491	1.070802121	0.417106669	0.506839054	0.615569023	0.485612105
9	[-0.08933853983005413, 0.9168321518824785, 1.2075776892672374]	8	-0.16233944	0.923591881	1.064975544	0.361824383	0.676950258	0.679214926	0.503485882
10	[-0.08933853983005413, 0.9168321518824785, 1.2075776892672374]	9	-0.207812502	0.926600738	1.06030093	0.429983488	0.710207297	0.684419716	0.577812383
11	[-0.08933853983005413, 0.9168321518824785, 1.2075776892672374]	10	-0.270567461	0.930854126	1.057958587	0.473282111	0.743745848	0.801202541	0.376718838
12	[-0.08933853983005413, 0.9168321518824785, 1.2075776892672374]	11	-0.354173361	0.937046073	1.059737458	0.416674646	0.635893542	0.692571485	0.632440839
13	[-0.08933853983005413, 0.9168321518824785, 1.2075776892672374]	12	-0.462526022	0.945958038	1.06819184	0.401303214	0.718311139	0.663890612	0.552066346
14	[-0.08933853983005413, 0.9168321518824785, 1.2075776892672374]	13	-0.599498509	0.958590469	1.086652762	0.453482113	0.609394921	0.638694799	0.502910851
15	[-0.08933853983005413, 0.9168321518824785, 1.2075776892672374]	14	-0.768640338	0.976096544	1.119274652	0.371831648	0.627791692	0.634850038	0.452569523
16	[-0.08933853983005413, 0.9168321518824785, 1.2075776892672374]	15	-0.954835175	0.999872427	1.13422238	0.440351132	0.548331538	0.713126871	0.592208091
17	[-0.08933853983005413, 0.9168321518824785, 1.2075776892672374]	16	-1.136023282	1.027196537	1.142858415	0.426439822	0.782178184	0.790738971	0.539530198
18	[-0.08933853983005413, 0.9168321518824785, 1.2075776892672374]	17	-1.276204346	1.051087263	1.13492678	0.398921911	0.805381595	0.789015684	0.42904001
19	[-0.08933853983005413, 0.9168321518824785, 1.2075776892672374]	18	-1.428832768	1.080428469	1.142933516	0.392946082	0.668312624	0.687165029	0.551198785
20	[-0.08933853983005413, 0.9168321518824785, 1.2075776892672374]	19	-1.57027769	1.106987181	1.181505619	0.453122178	0.6744945	0.713198954	0.484560915

For the PID controller, we generated 70 rows of data.