Домашнее задание №6

6.1 Как было сказано на занятиях. Advantage функцию в PPO можно считать и учить по-разному. В задании предлагается написать и исследовать другой способ делать это. А именно использовать представление A(s,a) = r + \gamma V(s') - V(s), где s' - следующее состояние. То есть returns в данном случае использовать не нужно. Необходимо сравнить кривые обучения алгоритма с этим “новым” способом и “старым” способом (из практики) на задаче Pendulum.

Импорт необходимых библиотек.

In [1]:
```
import pandas as pd
import gym
import matplotlib.pyplot as plt
import numpy as np
from PPO import PPO
from PPO_hw1 import PPO_hw1
from PPO_hw2 import PPO_hw2
from PPO_hw3 import PPO_hw3
```

```
Warning: Gym version v0.24.0 has a number of critical issues with `gym.make` such that the `reset` and `step` fu
nctions are called before returning the environment. It is recommend to downgrading to v0.23.1 or upgrading to v
0.25.1
```

Алгоритм с практики реализован в PPO.py, а модифицированный алгоритм в соответствии с заданием 6.1 реализован в PPO_hw1.py

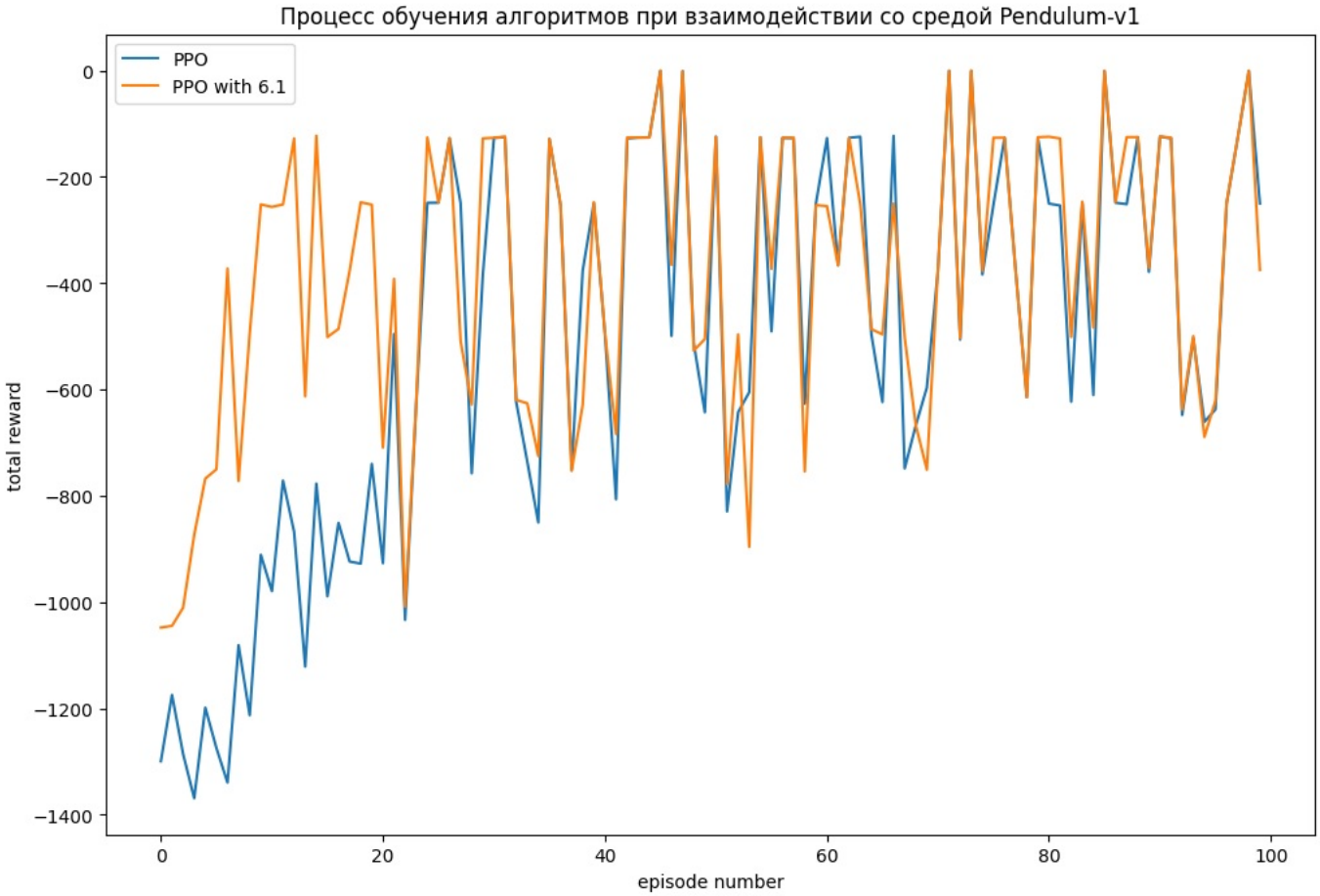
In [2]:
```
ppo = PPO(env=gym.make('Pendulum-v1'), n_episode=100, is_print=False)
ppo.fit()
```

```
/home/iukash/development/python/gym/gym_env/lib/python3.10/site-packages/gym/utils/env_checker.py:200: UserWarni
ng: WARN: We recommend you to use a symmetric and normalized Box action space (range=[-1, 1]) cf https://stable-
baselines3.readthedocs.io/en/master/guide/rl_tips.html
  logger.warn(
```

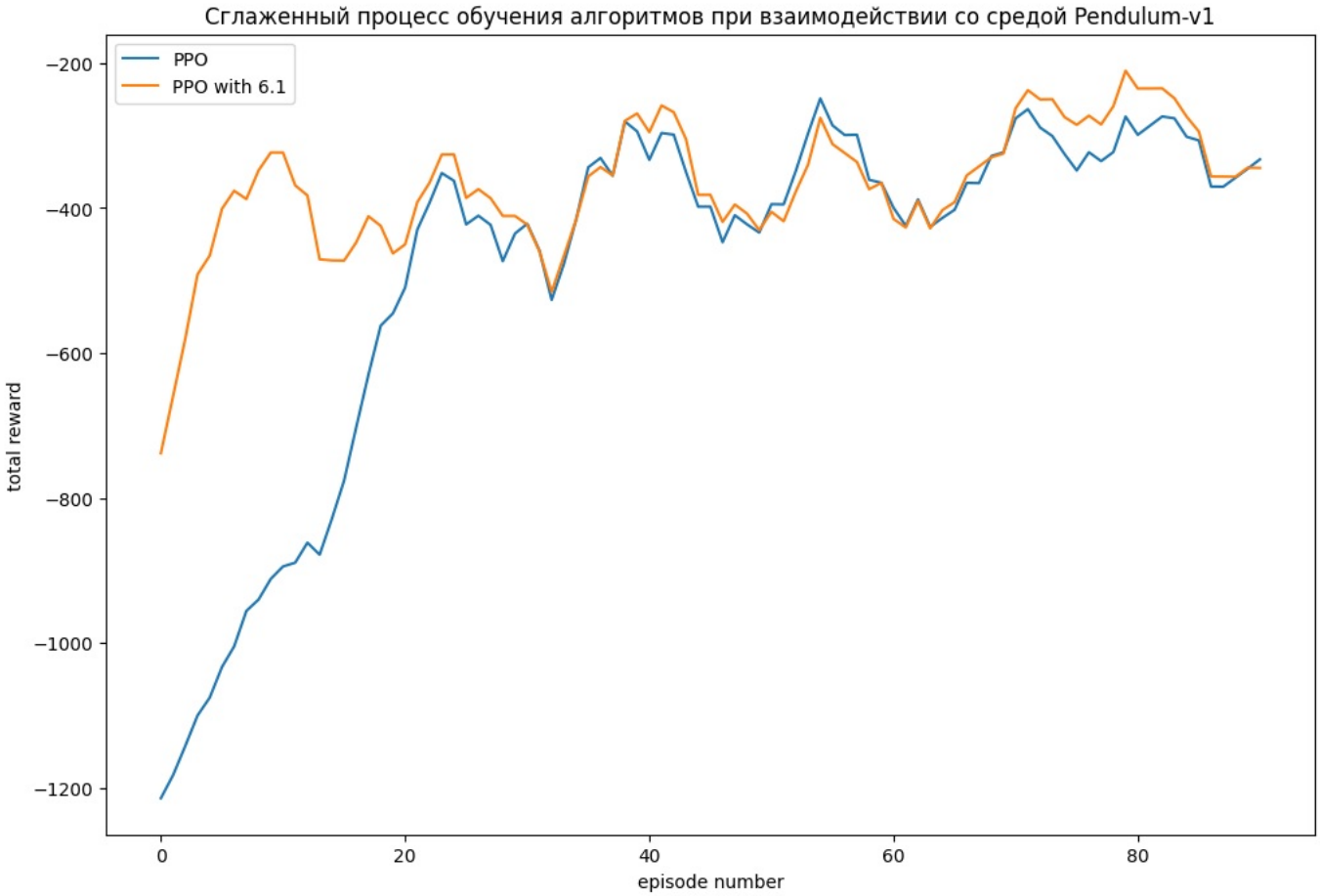
In [3]:
```
ppo_hw1 = PPO_hw1(env=gym.make('Pendulum-v1'), n_episode=100, is_print=False)
ppo_hw1.fit()
```

```
/home/iukash/development/python/gym/gym_env/lib/python3.10/site-packages/gym/utils/env_checker.py:200: UserWarni
ng: WARN: We recommend you to use a symmetric and normalized Box action space (range=[-1, 1]) cf https://stable-
baselines3.readthedocs.io/en/master/guide/rl_tips.html
  logger.warn(
```

In [4]:
```
plt.figure(figsize = (12, 8))
plt.plot(ppo.mean_total_rewards, label='PPO')
plt.plot(ppo_hw1.mean_total_rewards, label='PPO with 6.1')
plt.title('Процесс обучения алгоритмов при взаимодействии со средой Pendulum-v1')
plt.xlabel('episode number')
plt.ylabel('total reward')
plt.legend()
plt.show()
```



In [5]:
```
n = 10
ppo_rewards = pd.Series(ppo.mean_total_rewards).rolling(window=n).mean().iloc[n-1:].values
ppo_hw1_rewards = pd.Series(ppo_hw1.mean_total_rewards).rolling(window=n).mean().iloc[n-1:].values
plt.figure(figsize = (12, 8))
plt.plot(ppo_rewards, label='PPO')
plt.plot(ppo_hw1_rewards, label='PPO with 6.1')
plt.title('Сглаженный процесс обучения алгоритмов при взаимодействии со средой Pendulum-v1')
plt.xlabel('episode number')
plt.ylabel('total reward')
plt.legend()
plt.show()
```



Выводы по заданию 1:

Графики сходимости схожи, что показывает справедливость определения v-функции обоими способами.

6.2 На практике мы написали PPO для случая одномерного пространства действий. Использование же его для многомерного пространства действий требует небольших технических изменений в коде (при этом содержательно ничего не меняется). Задание заключается в том, чтобы внести эти изменения (т.е. модифицировать PPO для работы в средах с многомерным пространством действий) и решить с его помощью LunarLander (результат должен быть больше 100). Для того, чтобы сделать LunarLander с непрерывным пространством действий нужно положить continuous=True (см. пояснения в Lunar Lander - Gym Documentation (gymlibrary.dev))

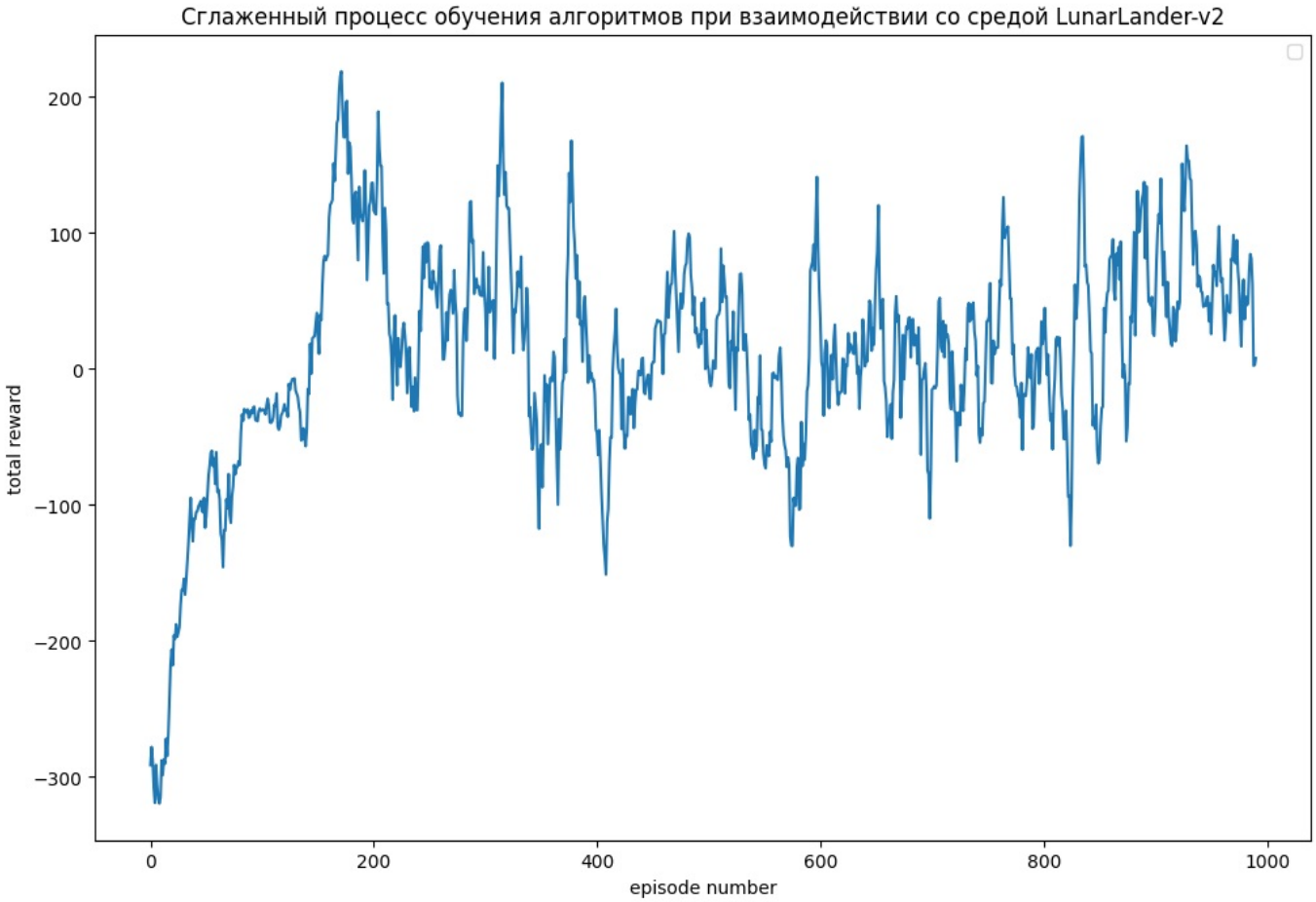
Обобщение алгоритма на взаимодействие с многомерным пространством действий реализовано в PPO_hw2.py.

In [2]:
```
%%time
ppo_hw2 = PPO_hw2(env=gym.make('LunarLander-v2', continuous=True), max_len_trajectory=1000, n_episode=1000,
                  n_trajectory=50, n_neurons=256, epoch_n=5, is_print=False)
ppo_hw2.fit()
```

```
CPU times: user 1d 2h 53min 21s, sys: 1min 35s, total: 1d 2h 54min 57s
Wall time: 6h 52min 5s
```

In [12]:
```
n = 10
ppo_hw2_rewards = pd.Series(ppo_hw2.mean_total_rewards[:1000]).rolling(window=n).mean().iloc[n-1:].values
plt.figure(figsize = (12, 8))
plt.plot(ppo_hw2_rewards)
plt.title('Сглаженный процесс обучения алгоритмов при взаимодействии со средой LunarLander-v2')
plt.xlabel('episode number')
plt.ylabel('total reward')
plt.legend()
plt.show()
```

```
No artists with labels found to put in legend.  Note that artists whose label start with an underscore are ignor
ed when legend() is called with no argument.
```



Выводы по заданию 2:

Особых проблем с кодом не было. Однако среда LunarLander-v2 требовательна к подбору гиперпараметров, в первую очередь количество итераций.

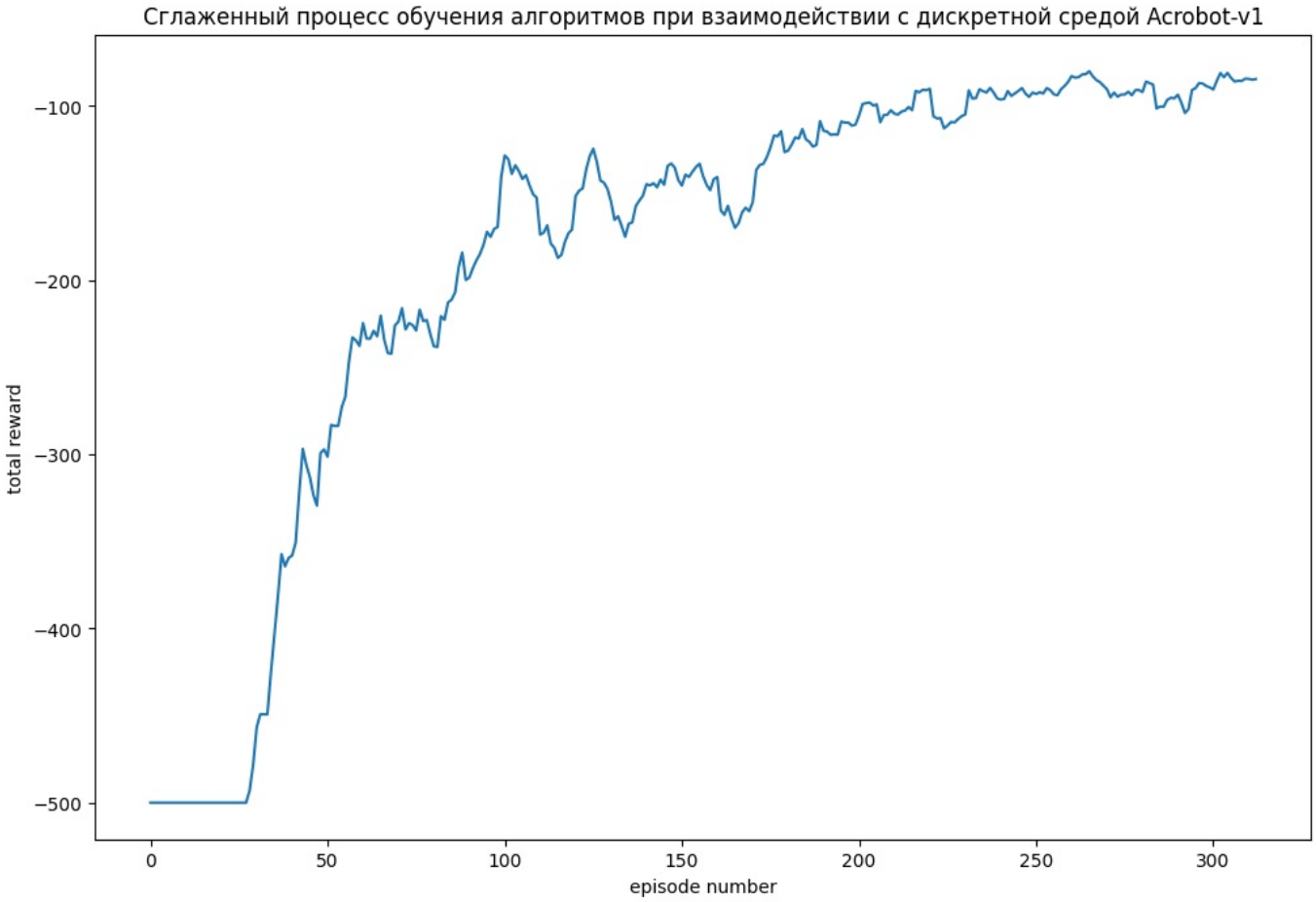
6.3 Написать PPO для работы в средах с конечным пространствам действий и решить Acrobot. Для решения можно использовать Categorial из torch.distributions (см. pytorch документацию).

Класс обертка над средой в целях перевода значения действия от -1 до 1.

In [4]:
```
class AcrobotActionWrapper(gym.ActionWrapper):
    """Change the action range (0, 2) to (-1, 1)."""
    def action(self, action):
        if action == 0:
            return -1
        elif action == 1:
            return 0
        elif action == 2:
            return 1
        return action
```

In [ ]:
```
ppo_hw3 = PPO_hw3(env=AcrobotActionWrapper(gym.make('Acrobot-v1')), max_len_trajectory=500, n_episode=300, n_tra
                  n_neurons=128, epoch_n=5, is_print=False)
ppo_hw3.fit()
```

In [20]:
```
n = 10
ppo_hw3_rewards = pd.Series(ppo_hw3.mean_total_rewards).rolling(window=n).mean().iloc[n-1:].values
plt.figure(figsize = (12, 8))
plt.plot(ppo_hw3_rewards)
plt.title('Сглаженный процесс обучения алгоритмов при взаимодействии с дискретной средой Acrobot-v1')
plt.xlabel('episode number')
plt.ylabel('total reward')
plt.show()
```



Выводы по заданию 3:

Без класса-обертки добится нормальной сходимости так и не получилось и тоже требует достаточно большого количества итераций (в отличии от DQN), поэтому для каждой задачи в зависимости от условий подходит соответствующий алгоритм.

In [ ]: