```
import gym
import numpy as np
import random

env = gym.make("FrozenLake-v0", map_name='4x4', is_slippery=False)
```

Funkcja generująca politykę stochastyczną:

```
def create_random_sto_policy(env):
    policy = {}
    for key in range(0, env.observation_space.n):
        p = {}
        for action in range(0, env.action_space.n):
            p[action] = 1 / env.action_space.n
        policy[key] = p
    return policy
```

Testujemy:

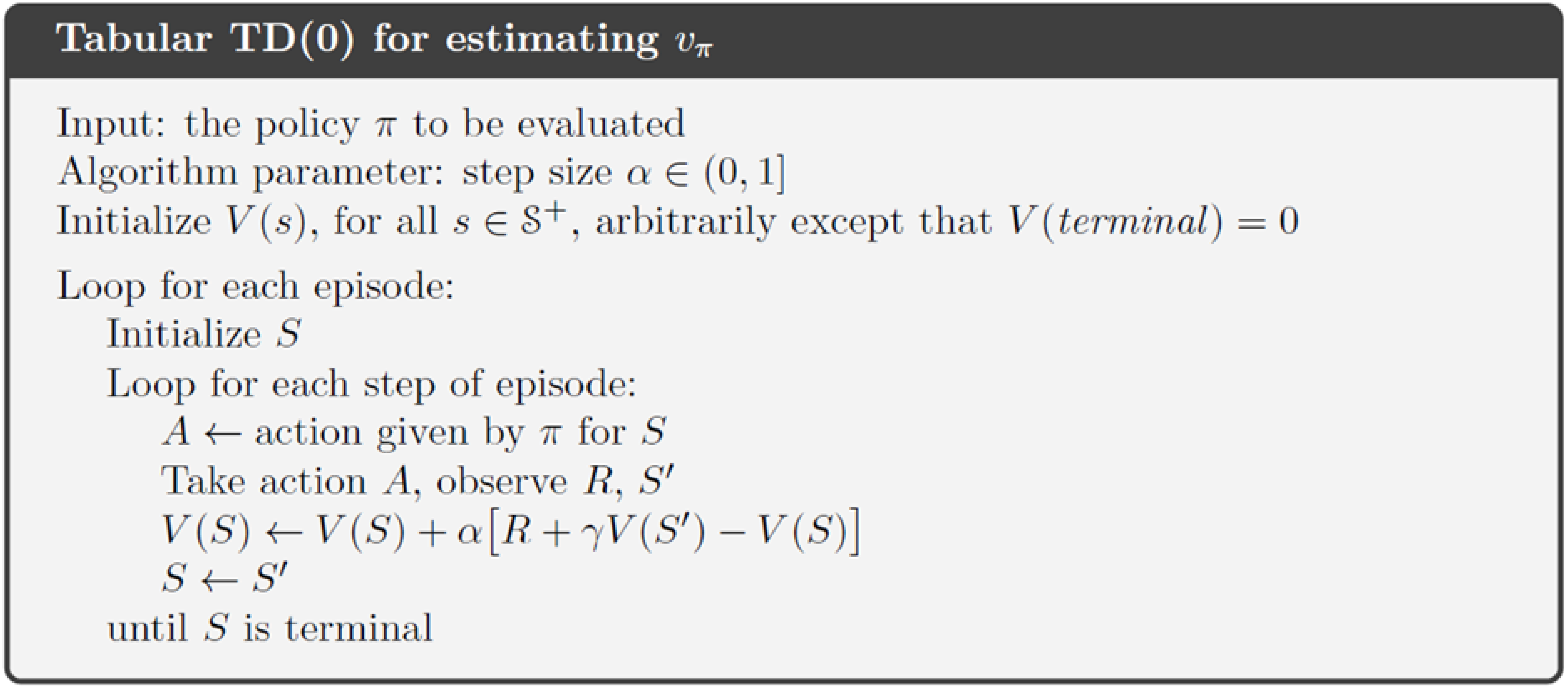
```
policy = create_random_sto_policy(env)
policy
```

```
     {0: {0: 0.25, 1: 0.25, 2: 0.25, 3: 0.25},
      1: {0: 0.25, 1: 0.25, 2: 0.25, 3: 0.25},
      2: {0: 0.25, 1: 0.25, 2: 0.25, 3: 0.25},
      3: {0: 0.25, 1: 0.25, 2: 0.25, 3: 0.25},
      4: {0: 0.25, 1: 0.25, 2: 0.25, 3: 0.25},
      5: {0: 0.25, 1: 0.25, 2: 0.25, 3: 0.25},
      6: {0: 0.25, 1: 0.25, 2: 0.25, 3: 0.25},
      7: {0: 0.25, 1: 0.25, 2: 0.25, 3: 0.25},
      8: {0: 0.25, 1: 0.25, 2: 0.25, 3: 0.25},
      9: {0: 0.25, 1: 0.25, 2: 0.25, 3: 0.25},
      10: {0: 0.25, 1: 0.25, 2: 0.25, 3: 0.25},
      11: {0: 0.25, 1: 0.25, 2: 0.25, 3: 0.25},
      12: {0: 0.25, 1: 0.25, 2: 0.25, 3: 0.25},
      13: {0: 0.25, 1: 0.25, 2: 0.25, 3: 0.25},
      14: {0: 0.25, 1: 0.25, 2: 0.25, 3: 0.25},
      15: {0: 0.25, 1: 0.25, 2: 0.25, 3: 0.25}}
```

Wybór akcji dla stanu s (1,...,15):

```
s=5
n = random.uniform(0,1)
top_range = 0
for prob in policy[s].items():
  top_range += prob[1]
  if n < top_range:
    action = prob[0]
    break

print(action)
```

```
     2
```



Tabular TD(0) for estimating v_π

```
Input: the policy $\pi$ to be evaluated
Algorithm parameter: step size $\alpha \in (0, 1]$
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop for each episode:
    Initialize $S$
    Loop for each step of episode:
        $A \leftarrow$ action given by $\pi$ for $S$
        Take action $A$, observe $R$, $S'$
        $V(S) \leftarrow V(S) + \alpha\big[R + \gamma V(S') - V(S)\big]$
        $S \leftarrow S'$
    until $S$ is terminal
```

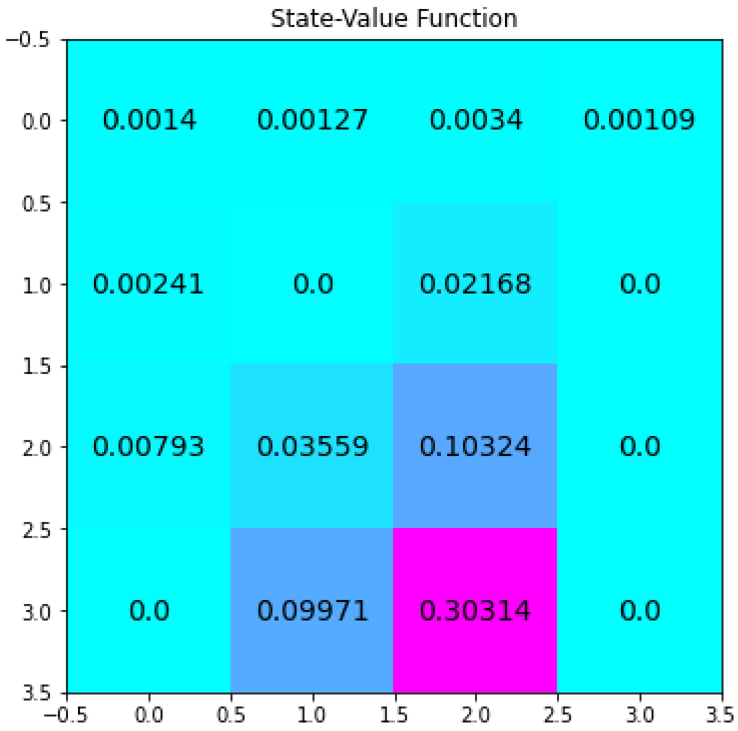
Funkcja wyliczająca V (do uzupełnienia):

```
def TD_0(env, episodes=1000, gamma=0.9, alpha=0.4):

    V = np.zeros(env.nS)

    policy = create_random_sto_policy(env)

    for i in range(episodes):

        finished = False
        env.reset()
        s = env.s

        while not finished:
         n = random.uniform(0,1)
         top_range = 0
         for prob in policy[s].items():
          top_range += prob[1]
          if n < top_range:
           action = prob[0]
           break
         next_s,R,finished,_= env.step(action)

         V[s] += policy[s][action] * alpha * (R + gamma * V[next_s]- V[s])

         s = next_s
         if finished:
          break
         # update value for none terminal states
         #else:
         # V[s] += V[s] +  alpha* (R + gamma * V[next_s] - V[s])


    return V
```

Testujemy:

```
V = TD_0(env,episodes=500)
```

```
from plot_utils import plot_values   #konieczne wczytanie pliku 'plot_utils.py'
plot_values(V)
```


State-Value Function
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
0.0014
0.00127
0.0034
0.00109
0.00241
0.0
0.02168
0.0
0.00793
0.03559
0.10324
0.0
0.0
0.09971
0.30314
0.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5

✓ 0 s ukończono o 13:16 ● ✕