Hill Climbing Algorithm

In [1]:

```
import francium.algorithms.hill_climbing as hc
import francium.core.eval_functions as eval_functions
from francium.core import State
```

using an environment with z = x^2 + y^2

In [2]:

```
agent = hc.Agent(step_size=1e-1)
env = hc.Environment(x_bounds=(-5.0, 5.0), y_bounds=(-5.0, 5.0), eval_func=eval_functions.convex_x_square)
solver = hc.Solver(agent=agent, environment=env)
```

In [3]:

```
solver.init_solver(
    init_state=State({
        'x': 4.0,
        'y': 2.0,
        'z': env.evaluation_func(4.0, 2.0)
    })
)
```

```
[ 2020-12-06 19:56:15,109 - francium.algorithms.hill_climbing.solver ] INFO: => Initialized Solver with State: {'x': 4.0, 'y':
2.0, 'z': 20.0}
```

In [4]:

```
for episode in range(1000):
    trainable = solver.train_step()
    if not trainable:
        break
```

In [5]:

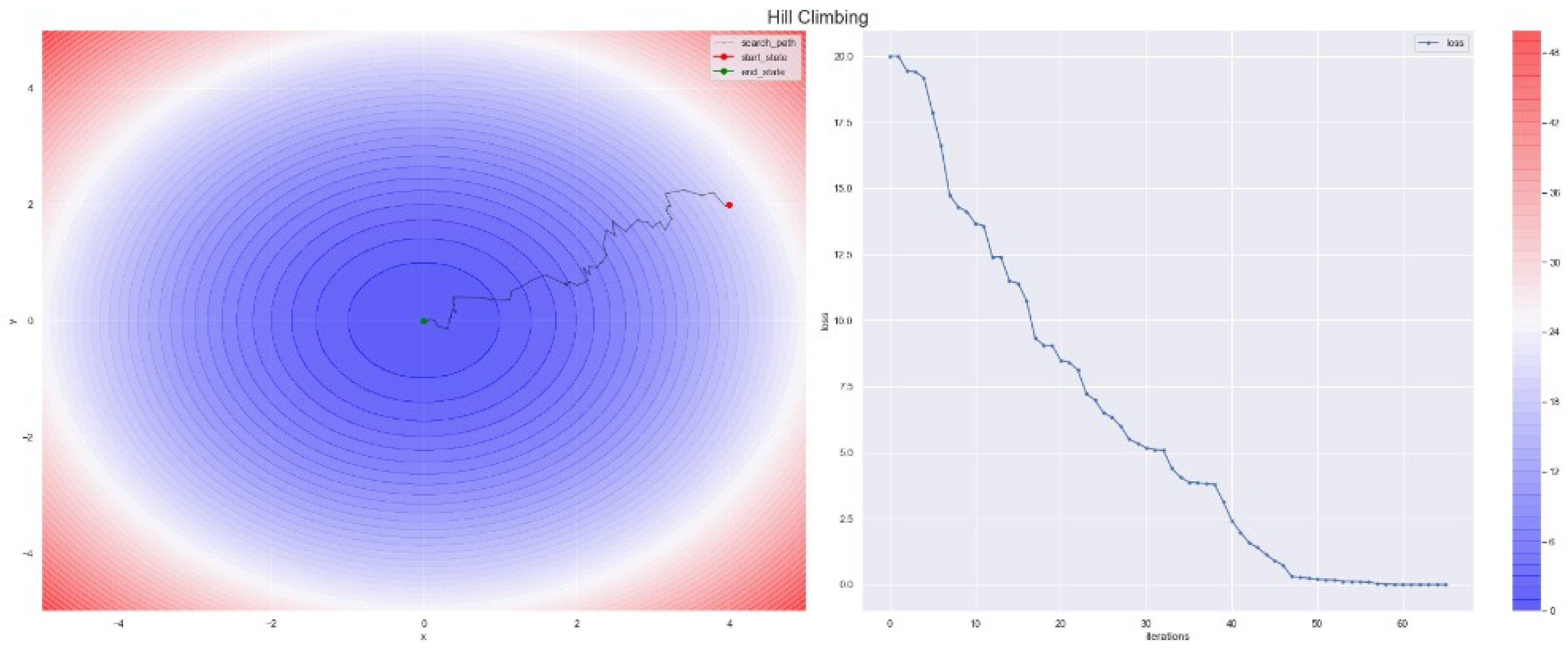
```
solver.memory.best_episode
```

Out[5]:

```
{'x': -0.003999626443947791, 'y': -0.004717388353104505, 'z': 3.825076911463814e-05}
```

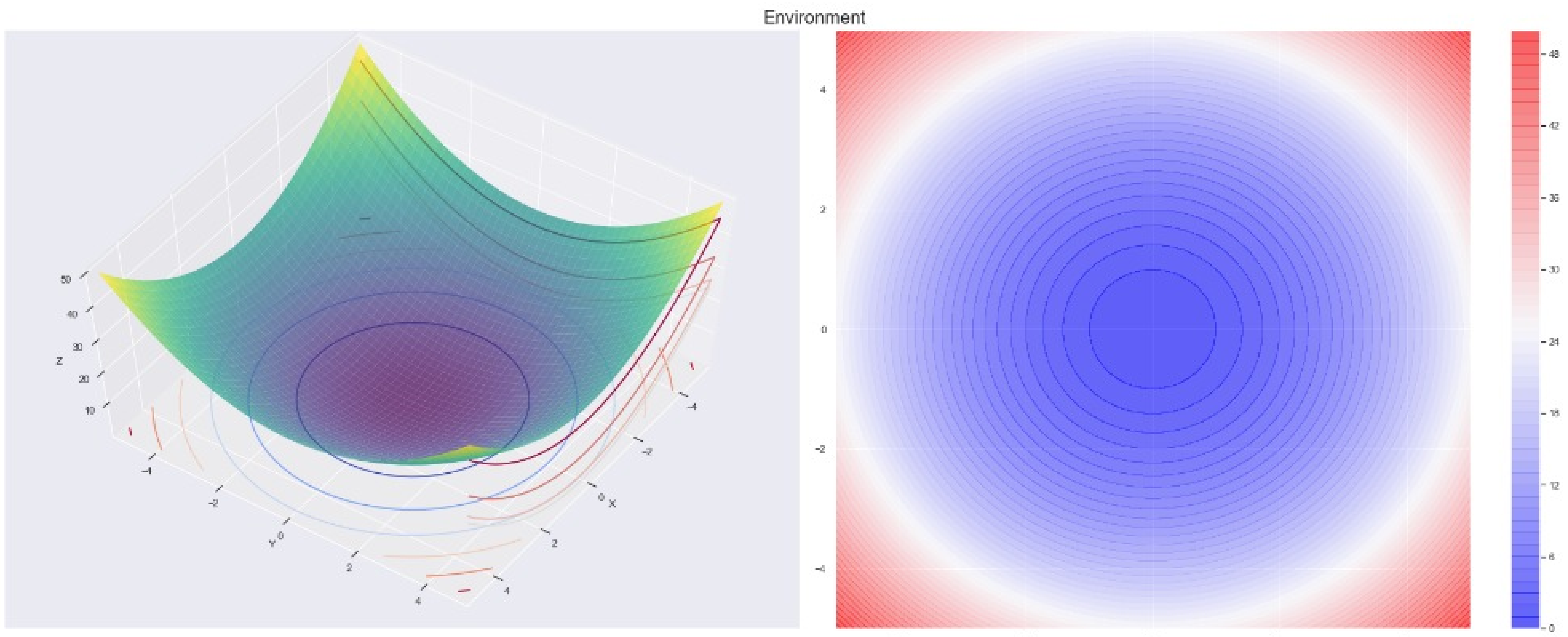
In [6]:

```
solver.plot_history()
```



In [7]:

```
env.plot_environment()
```



using an environment with z = 5 * sin(x^2 + y^2) + x^2 + y^2

In [8]:

```
agent = hc.Agent(step_size=1e-1)
env = hc.Environment(x_bounds=(-5.0, 5.0), y_bounds=(-5.0, 5.0), eval_func=eval_functions.sinx_plus_x)
solver = hc.Solver(agent=agent, environment=env)
```

In [9]:

```
solver.init_solver(
    init_state=State({
        'x': 4.0,
        'y': 2.0,
        'z': env.evaluation_func(4.0, 2.0)
    })
)
```

```
[ 2020-12-06 19:56:16,715 - francium.algorithms.hill_climbing.solver ] INFO: => Initialized Solver with State: {'x': 4.0, 'y':
2.0, 'z': 24.56472625363814}
```

In [10]:

```
for episode in range(1000):
    trainable = solver.train_step()
    if not trainable:
        break
```

In [11]:

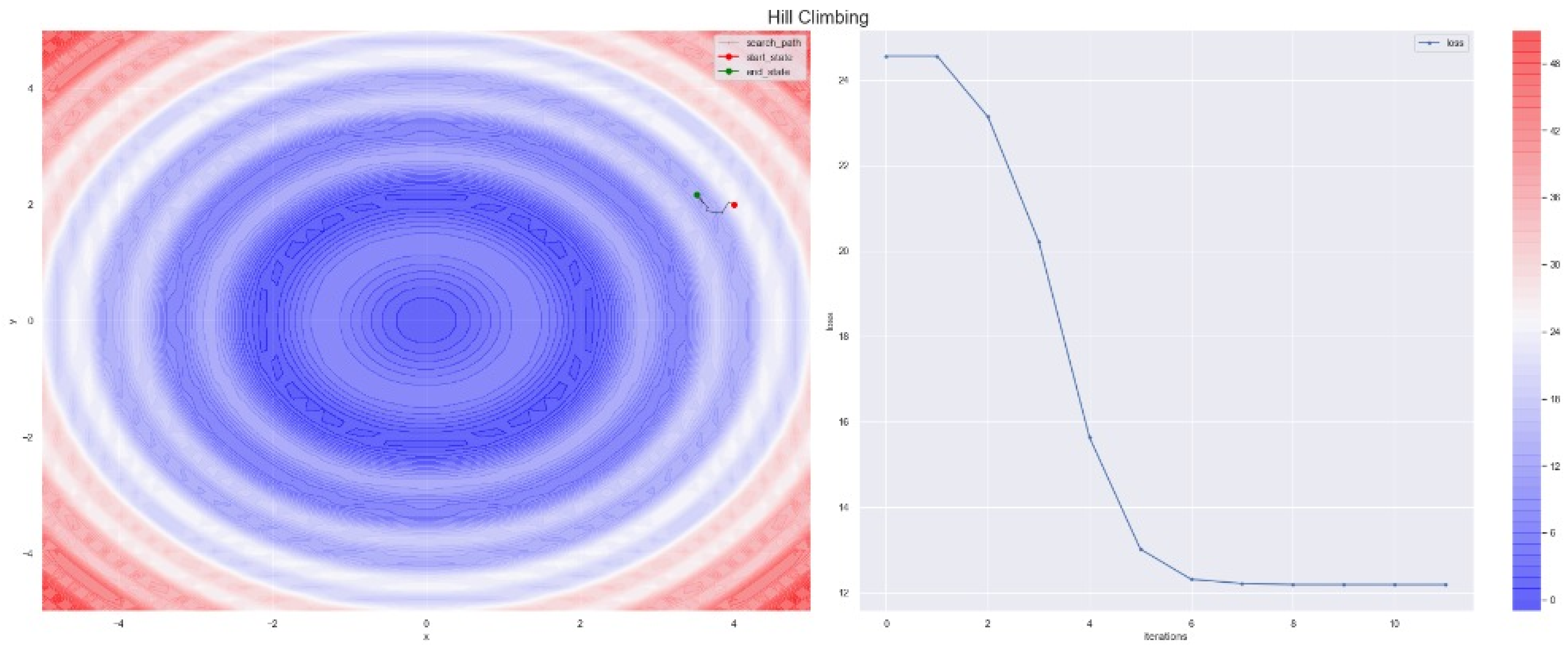
```
solver.memory.best_episode
```

Out[11]:

```
{'x': 3.5189357492297706, 'y': 2.1670941140655553, 'z': 12.178430161309148}
```

In [12]:

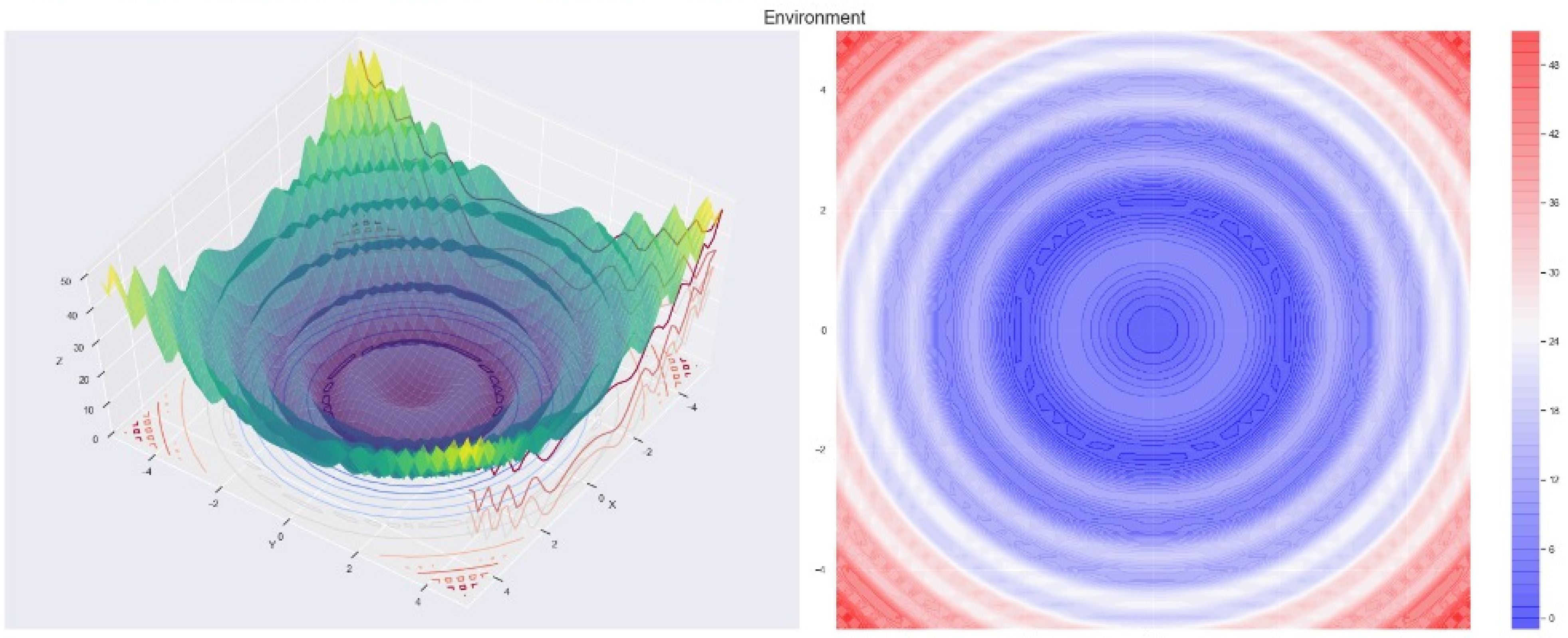
```
solver.plot_history()
```



In [13]:

```
env.plot_environment()
```

```
C:\Users\shadowleaf\anaconda3\envs\thetensorclan-aws\lib\site-packages\numpy\core\_asarray.py:136: VisibleDeprecationWarning: C
reating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths
or shapes) is deprecated. If you meant to do this, you must specify 'dtype=object' when creating the ndarray
  return array(a, dtype, copy=False, order=order, subok=True)
```



In [13]: