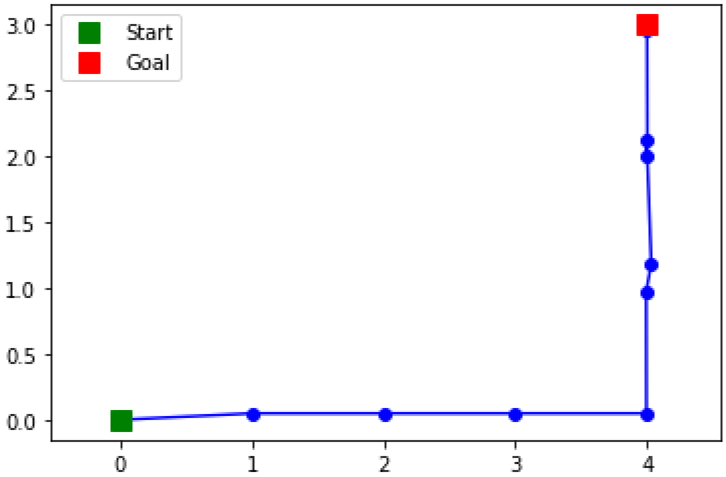
```
### Link to Github Repo with the same code: ###
### https://github.com/dylan-losey/me5824.git ###

import matplotlib.pyplot as plt
import numpy as np
from scipy.optimize import minimize, LinearConstraint, NonlinearConstraint


### Create a class to perform the trajectory optimization ###

class TrajOpt(object):

    def __init__(self):

        # initialize trajectory
        self.n_waypoints = 10
        self.n_dof = 2
        self.home = np.array([0., 0.])
        self.xi0 = np.zeros((self.n_waypoints, self.n_dof))
        self.xi0 = self.xi0.reshape(-1)

        # create start constraint and action constraint
        self.B = np.zeros((self.n_dof, self.n_dof * self.n_waypoints))
        for idx in range(self.n_dof):
            self.B[idx,idx] = 1
        self.lincon = LinearConstraint(self.B, self.home, self.home)
        self.nonlincon = NonlinearConstraint(self.nl_function, -1.0, 1.0)

    # each action cannot move more than 1 unit
    def nl_function(self, xi):
        xi = xi.reshape(self.n_waypoints, self.n_dof)
        actions = xi[1:, :] - xi[:-1, :]
        return np.linalg.norm(actions, axis=1)

    # trajectory cost function
    def trajcost(self, xi):
        xi = xi.reshape(self.n_waypoints, self.n_dof)
        cost = 0
        ### define your cost function here ###
        ### here is an example encouraging the robot to reach [5, 2] ###
        for idx in range(self.n_waypoints):
            cost += np.linalg.norm(np.array([4., 3.]) - xi[idx, :])
            cost += 1*abs(xi[idx, 1])
            if xi[idx, 1] < 0.05:
              cost -= 1*xi[idx, 1]
        return cost

    # run the optimizer
    def optimize(self):
        res = minimize(self.trajcost, self.xi0, method='SLSQP', constraints={self.lincon, self
        xi = res.x.reshape(self.n_waypoints, self.n_dof)
        return xi, res


### Run the trajectory optimizer ###
```

```
trajopt = TrajOpt()
xi, res = trajopt.optimize()
print(xi)
plt.plot(xi[:,0], xi[:,1], 'bo-')
plt.plot(0, 0,  'gs', markersize=10, label='Start')
plt.plot(4, 3,  'rs', markersize=10, label='Goal')
plt.legend()
plt.axis("equal")
plt.show()
```

```
[[0.         0.        ]
 [0.99879898 0.0489958 ]
 [1.99879891 0.04901323]
 [2.99879884 0.04899491]
 [3.99387065 0.04898458]
 [3.99188275 0.97197579]
 [4.03034504 1.18648881]
 [3.99522503 2.13092249]
 [4.00148332 1.99750678]
 [3.99855789 2.96269541]]
```


🟩 Start
🟥 Goal