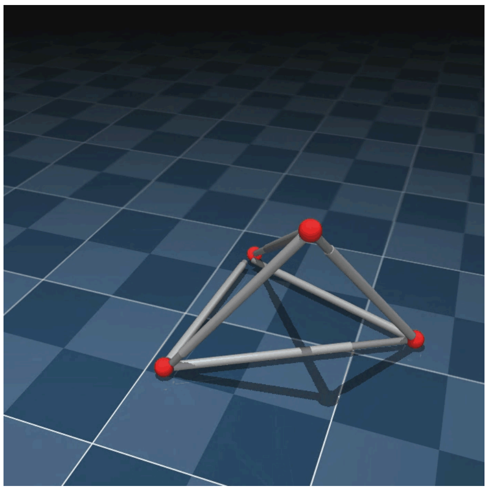
Tetrahedron



This environment is part of the Mujoco environments which contains general information about the environment.

Action Space	`Box(-1.0, 1.0, (6,), float32)`
Observation Space	`Box(-inf, inf, (16,), float64)`
import	`gymnasium.make("Tetrahedron-v1")`

Description

This environment consist of a tetrahedron truss robot where the each edge (**member**) of the tetrahedron is a linear actuator and the vertices (**nodes**) are the passive joints. The robot is controlled by changing the length of the linear actuators either dynamically or by changing the position of the center of mass of the robot. The goal of the robot is to reach the target position.

Notes

Problem parameters:

- n : number of nodes
- m : number of members
- l_i : length of the ith member ($i \in \{1, 2, \cdots, m\}$)
- (x_i, y_i, z_i) : coordinates of the ith node ($i \in \{1, 2, \cdots, n\}$)
- (x_{cm}, y_{cm}, z_{cm}) : coordinates of the center of mass
- $(v_{cm_x}, v_{cm_y}, v_{cm_z})$: velocity of the center of mass

Action Space

The action space is a `Box(0,2,(6,), float32)`. An action is a vector of 6 elements where each element is the length of the member. The length of the member is between 0 and 2 meters.

Num	Action	Min	Max	Name	Joint	Unit
0	position	0	2	[Member_1]	slide	lenght (m)
1	position	0	2	[Member_2]	slide	lenght (m)
2	position	0	2	[Member_3]	slide	lenght (m)
3	position	0	2	[Member_4]	slide	lenght (m)
4	position	0	2	[Member_5]	slide	lenght (m)
5	position	0	2	[Member_6]	slide	lenght (m)

notes: might need to change the position control to force control, thus we can have force as an input and lenght as an observation.

Observation Space

Observations capture the positional values of the center of mass and the respective time derivative aka velocity of the center of mass. Additionally, the positional values of the nodes are also captured. The observation space is a `Box(-inf, inf, (18,), float64)`. We might need to alter the observation space to have a better learning performance.

Num	Observation	Min	Max	Name	Joint	Unit
0	x-coord of CoM	-inf	inf	coord_x	free	(m)
1	y-coord of CoM	-inf	inf	coord_y	free	(m)
2	z-coord of CoM	-inf	inf	coord_z	free	(m)
3	x-linvel of CoM	-inf	inf	vel_x	free	(m/s)
4	y-linvel of CoM	-inf	inf	vel_y	free	(m/s)
5	z-linvel of CoM	-inf	inf	vel_z	free	(m/s)
6	length of [M_1]	-inf	inf	length_1	slide	(m)

7	length of [M_2]	-inf	inf	length_2	slide	(m)
8	length of [M_3]	-inf	inf	length_3	slide	(m)
9	length of [M_4]	-inf	inf	length_4	slide	(m)
10	length of [M_5]	-inf	inf	length_5	slide	(m)
11	length of [M_6]	-inf	inf	length_6	slide	(m)
12	x-coord of [N_1]	-inf	inf	coord_x	free	(m)
13	y-coord of [N_1]	-inf	inf	coord_y	free	(m)
14	z-coord of [N_1]	-inf	inf	coord_z	free	(m)
15	x-coord of [N_2]	-inf	inf	coord_x	free	(m)
16	y-coord of [N_2]	-inf	inf	coord_y	free	(m)
17	z-coord of [N_2]	-inf	inf	coord_z	free	(m)
18	x-coord of [N_3]	-inf	inf	coord_x	free	(m)
19	y-coord of [N_3]	-inf	inf	coord_y	free	(m)
20	z-coord of [N_3]	-inf	inf	coord_z	free	(m)
21	x-coord of [N_4]	-inf	inf	coord_x	free	(m)
22	y-coord of [N_4]	-inf	inf	coord_y	free	(m)
23	z-coord of [N_4]	-inf	inf	coord_z	free	(m)

- add node positions
- add ground nodes
- add member orientation

notes: for the dynamic movement, linear forces rather than positions are controlled.
notes: might want to minimize the lengths at the end, or final configuration enforces that.

Rewards

- `healthy_reward` : The robot is healthy and the target position is not reached.

- *forward_reward* : A reward of moving forward in the x-direction which is measured as (x-coordinate before action - x-coordinate after action)/dt. dt is the time between actions and is dependent on the `frame_skip` parameter (default is 5), where the frametime is 0.01 - making the default dt = 5 * 0.01 = 0.05. This reward would be positive if the robot moves forward (in positive x direction).
- *ctrl_cost* : A penalty for the control effort. The control effort is the sum of the squares of the action values multiplied by the ctrl_cost_weight parameter (default is 0.05).
- *size_cost* : A penalty for the overall robot size. The size cost is the sum of the squares of the lengths of the members multiplied by the size_cost_weight parameter (default is 0.0001).

In the further implementation, the reward function might be changed to a more complex one. An example is the negative of the distance between the center of mass and the target position. The reward function is defined as:

$$r = -\sqrt{(x_{cm} - x_{target})^2 + (y_{cm} - y_{target})^2 + (z_{cm} - z_{target})^2}$$

another cost can be:

- *contact_cost* : A penalty for the contact forces. The contact forces are the sum of the squares of the forces on the robot multiplied by the `contact_cost_weight` parameter (default is 0.0001).

Episode Termination

The truss robot is said to be unhealthy if any of the following conditions are met:

- Any of the state space values are NaN.
- The z-coordinate of the center of mass i outside the range of $[0.1, 2]$ meters. (inflated)

The episode is terminated if the robot is unhealthy or the robot reaches the target position. trunckated at 1000 steps.