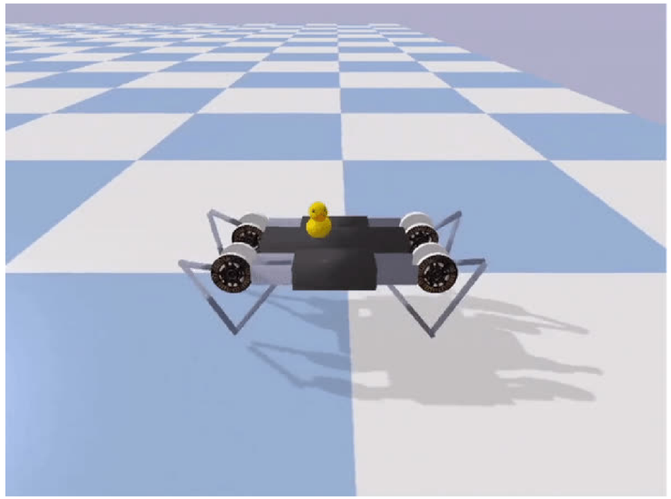
ROBOTICS CLUB
IIT (BHU) VARANASI

# PyBullet

Simplest Robotics Simulation Software



- Simulation is just a virtual rendering or very close proxy of the real world so that we can conduct experiments on our robot without having to build them in real life.
- Virtual experiments in simulations are less expensive and take less time than the actual experiments in real world.
- There are plenty of options available for simulation in robotics namely PyBullet, Gazebo, Webots, V-Rep, MujoCo, etc.

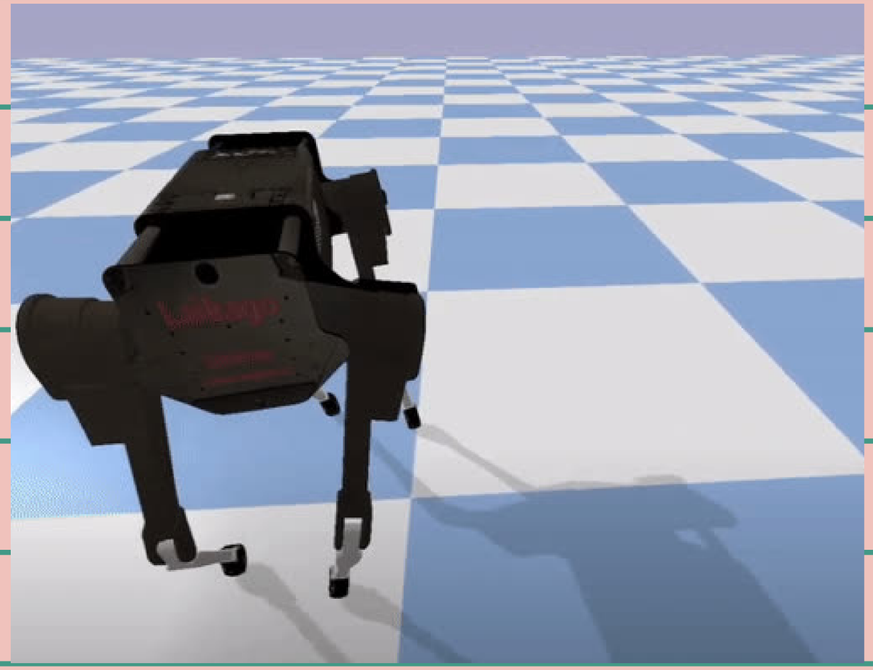
Why use simulations?

- Simulations are the best place to start when developing a new robot. By using a simulator to develop your robot, you can quickly identify if your idea is feasible or not with almost no expense. Additionally, you can easily test and discover which are the physical constraints that your robot must face to accomplish its goal.
- Simulators allow the easy and quick test of many different ideas for the same robotic problem, test them, and then decide which one to build based on actual data.
- Since your robot has been defined and tested in the simulator, you can start its physical construction. The good thing with simulators is that they allow you to keep doing tests even if your robot is not built yet.
- Bugs found in your robot software can be debugged first in the simulator.
- By debugging in the simulator you will save a lot of time since testing on the real robot is very time-consuming.
- Given that simulation is the way to go, there are plenty of options available for robotics namely Bullet, Gazebo, V-Rep, Webots, Open Dynamics Engine, MujoCo, etc.

PyBullet is a physics engine that simulates collision detection, soft and rigid body dynamic. It is an easy to use Python module for physics simulation, robotics, etc.

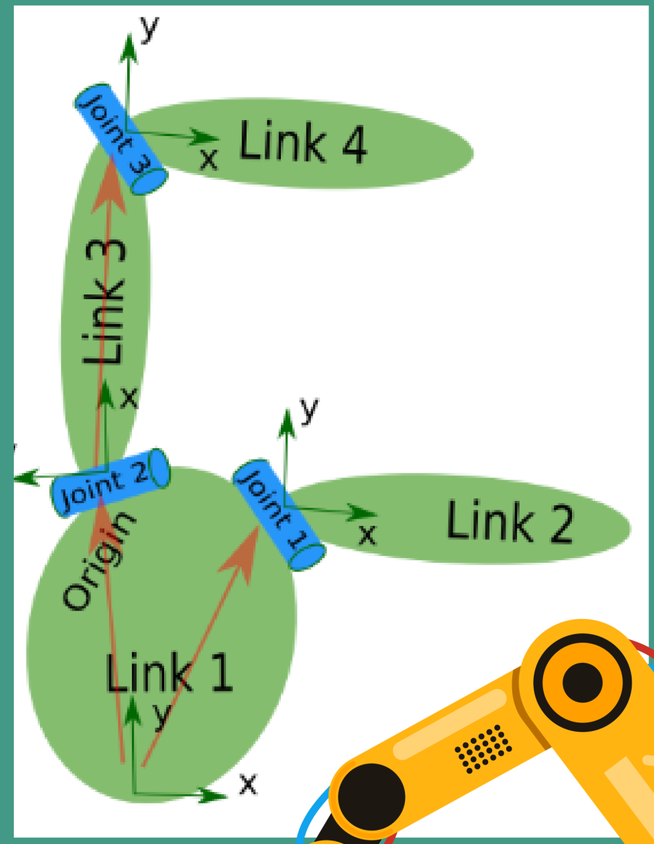
Why we choose PyBullet?

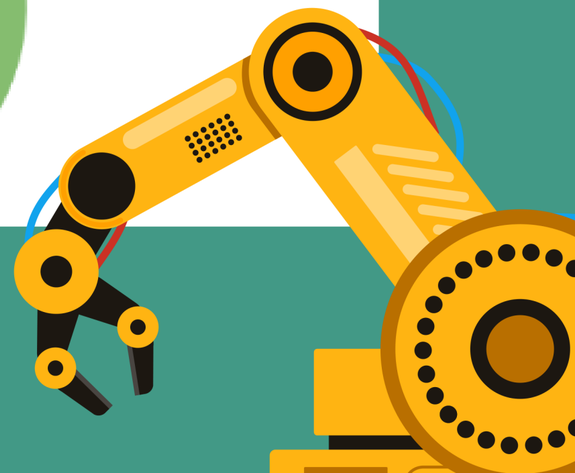
- It is an open source software with an active community.
- Built for python development, hence gives more informative and clear approach for beginners.
- We don’t require any external dependencies except a fully working python interpreter.



# Universal Robot Structure

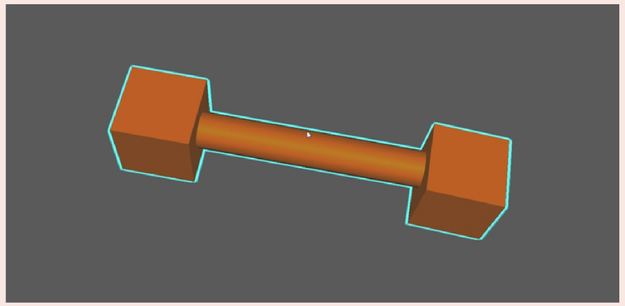
- So, a robot is basically any machine that, along with doing some computations, can move as well. And for that, in a robot, we have links - things that move - and joints - things that make the links move.
- A robot is made up of several links connected by movable joints. These joints have their respective motors that move these joints. Every robot has a base link, from which whole of the robot branches out.
- The base link is connected to a child link with some joint. That child link is connected to its own child link and acts as a parent link for its child link
- Joints: Any form of motion causing inter linkages are called as Joints. Joints are broadly classified into 6 types namely Fixed, Revolute, Continuous, Prismatic, Planar, Floating based on the range of motion it permit.



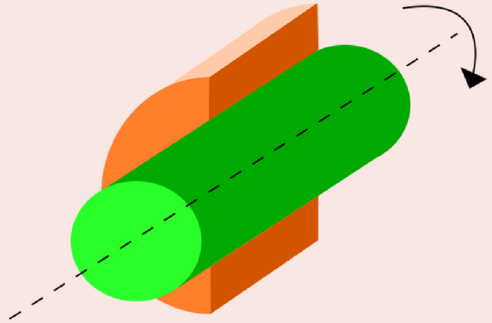



# Types of Joints

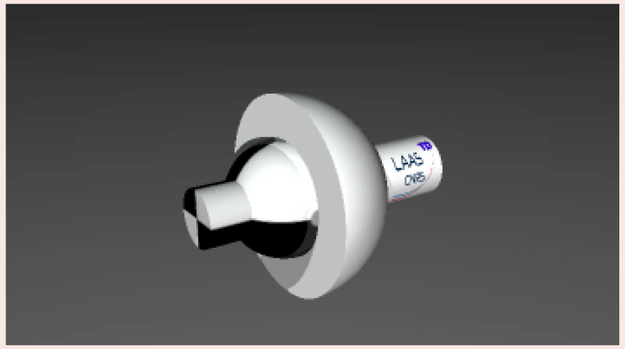
- Fixed Joints: These joints do not allow any type of relative motion between the two links.



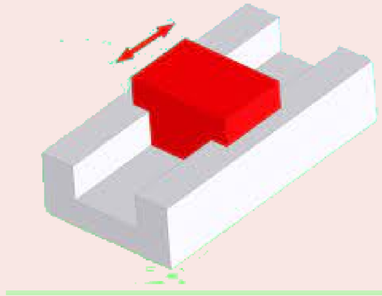
- Revolute Joints: Allow rotation about a single axis.



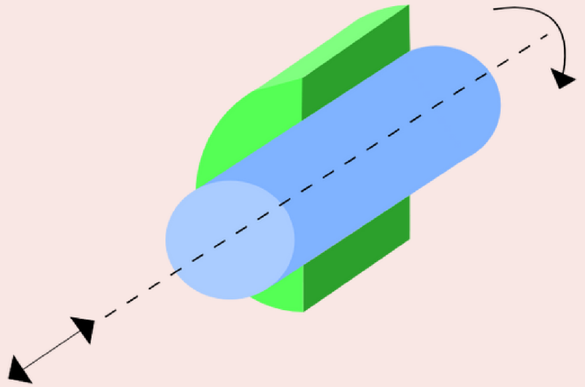
- Continuous Joints: Combination of Revolute Joints along all directions.



- Prismatic Joints: These Joints allow translation along a single axis.

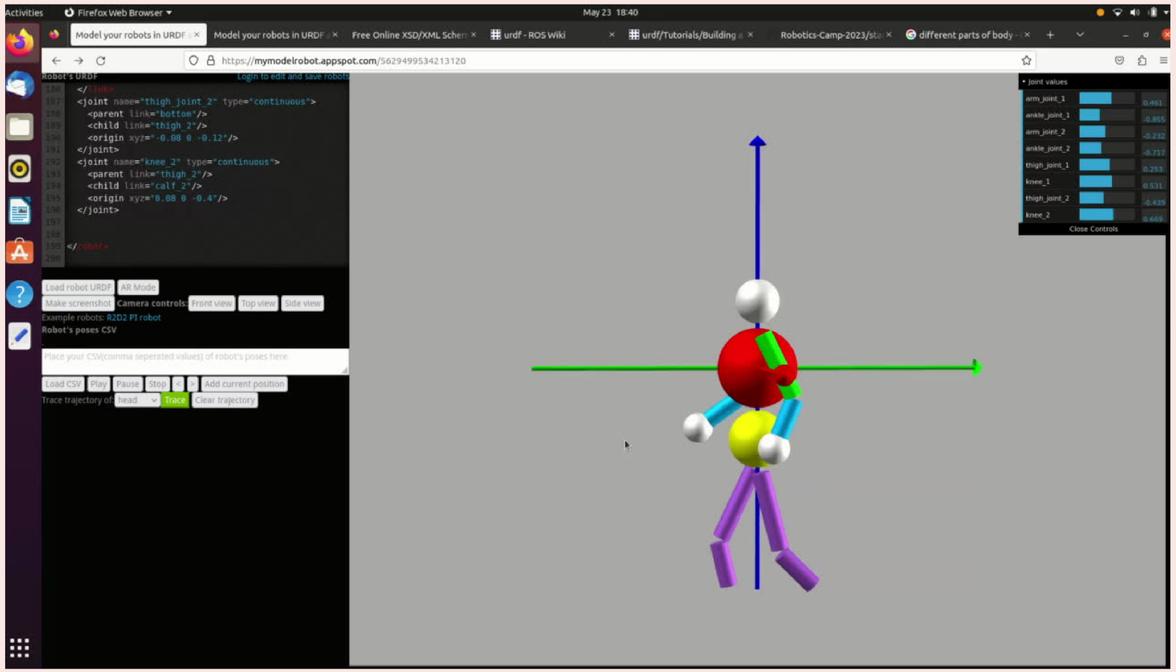


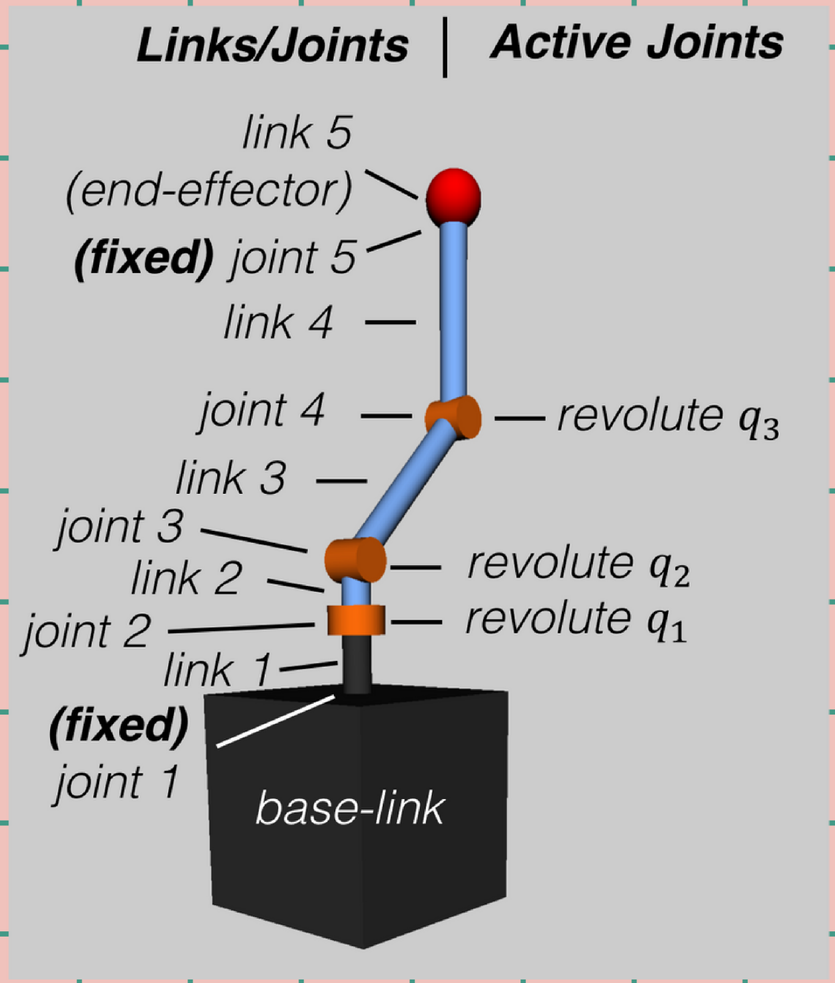
- Cylindrical Joints: Combination of Revolute and Prismatic Joints.



# URDF

The Unified Robotic Description Format (URDF) is a file that describes the robot properties like geometry, mass, inertia, collision model, etc in the form of tags which is easy to work with. It’s not just for the robot but for any object in simulation like a box, any obstacle, etc.



```python
import pybullet as p
import time
import pybullet_data
physicsClient = p.connect(p.GUI)#or p.DIRECT for non-graphical versi
p.setAdditionalSearchPath(pybullet_data.getDataPath()) #optionally
p.setGravity(0,0,-10)
planeId = p.loadURDF("plane.urdf")
cubeStartPos = [0,0,1]
cubeStartOrientation = p.getQuaternionFromEuler([0,0,0])
boxId = p.loadURDF("r2d2.urdf",cubeStartPos, cubeStartOrientation)
for i in range (10000):
    p.stepSimulation()
    time.sleep(1./240.)
cubePos, cubeOrn = p.getBasePositionAndOrientation(boxId)
print(cubePos,cubeOrn)
p.disconnect()
```

# Basic Functions

1. First we import the required libraries i.e. pybullet, time and pybullet_data
2. **connect**

- After importing the PyBullet module, the first thing to do is 'connecting' to the physics simulation. PyBullet is designed around a client-server driven API, with a client sending commands and a physics server returning the status. PyBullet has some built-in physics servers: DIRECT and GUI. Both GUI and DIRECT connections will execute the physics simulation and rendering in the same process as PyBullet.
- The connect function returns a physics client id.
- The DIRECT connection sends the commands directly to the physics engine, without using any transport layer and no graphics visualization window, and directly returns the status after executing the command.
- The GUI connection will create a new graphical user interface (GUI) with 3D OpenGL rendering, within the same process space as PyBullet.

3.setAdditionalSearchPath is used to add pybullet_data to the path which contains many examples, urdf files, etc.

4. setGravity:By default, there is no gravitational force enabled. setGravity lets you set the default gravity force for all objects. The setGravity input parameters are: (no return value):

| parameter type | Name | type | Description |
|---|---|---|---|
| required | gravityX | float | gravity force along the X world axis |
| required | gravityY | float | gravity force along the Y world axis |
| required | gravityZ | float | gravity force along the Z world axis |
| optional | physicsClientId | int | if you connect to multiple physics servers, you can pick which one. |

5.LoadURDF: The loadURDF will send a command to the physics server to load a physics model from a Universal Robot Description File (URDF).

We store the initial position of our urdf file in the variable cubsStartPos.

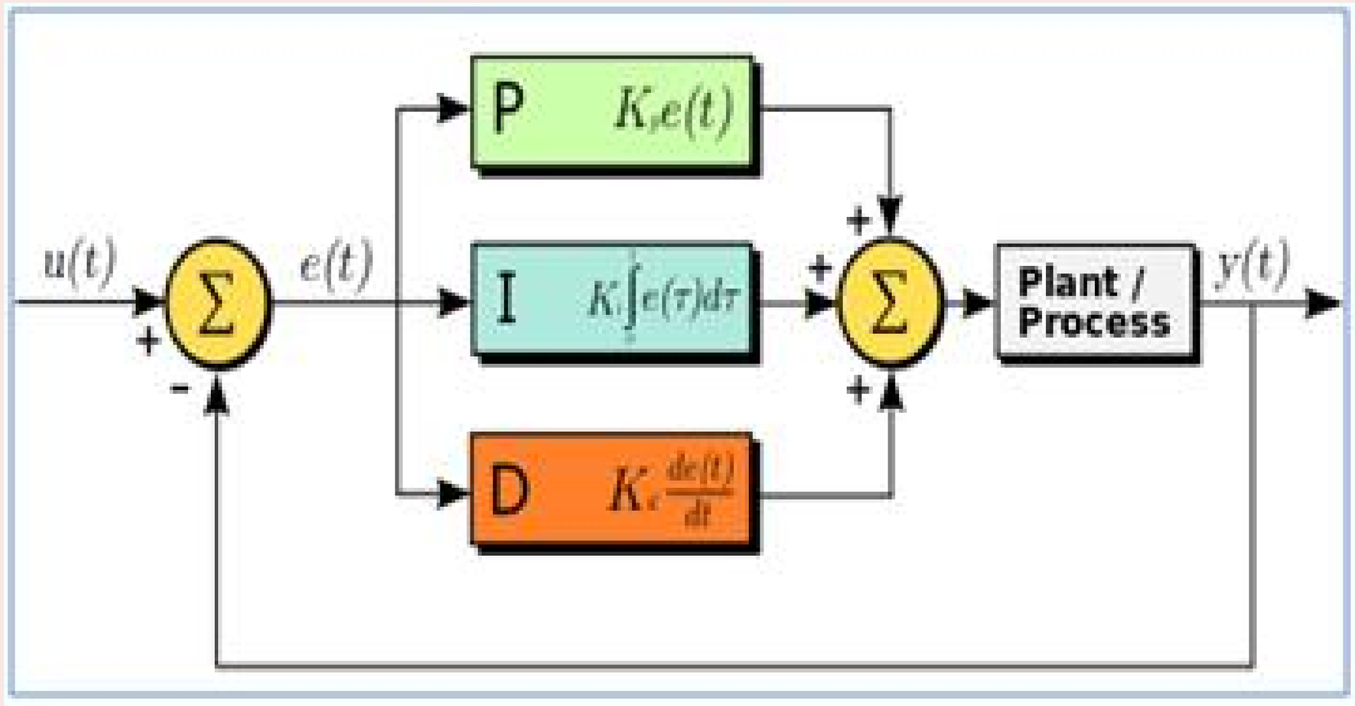
We store the initial Quaternion orientation of our urdf file in cubeStartOrientation (

We import our r2d2 robot urdf file in the desired position and orientation.

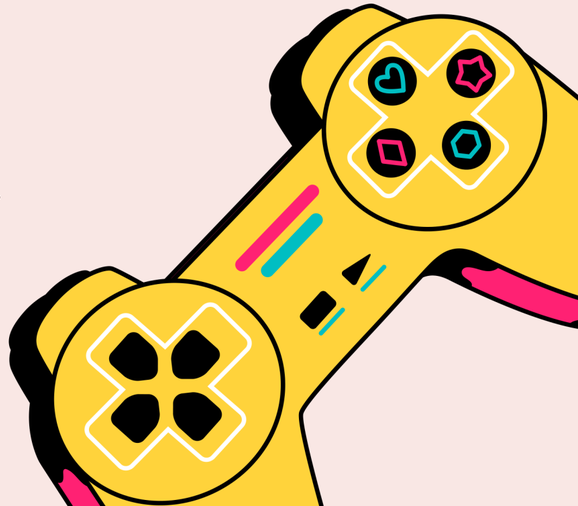
stepSimulation:stepSimulation will perform all the actions in a single forward dynamics simulation step. The default timestep is 1/240 second, it can be changed using the setTimeStep or setPhysicsEngineParameter API.

# PID Control !

- Robot control systems regulate and command the functions of the robot in order to achieve the desired result.




- The term PID stands for proportional integral derivative and it is one kind of device used to control different process variables like pressure, flow, temperature, and speed in industrial applications. In this controller, a control loop feedback device is used to regulate all the process variables.

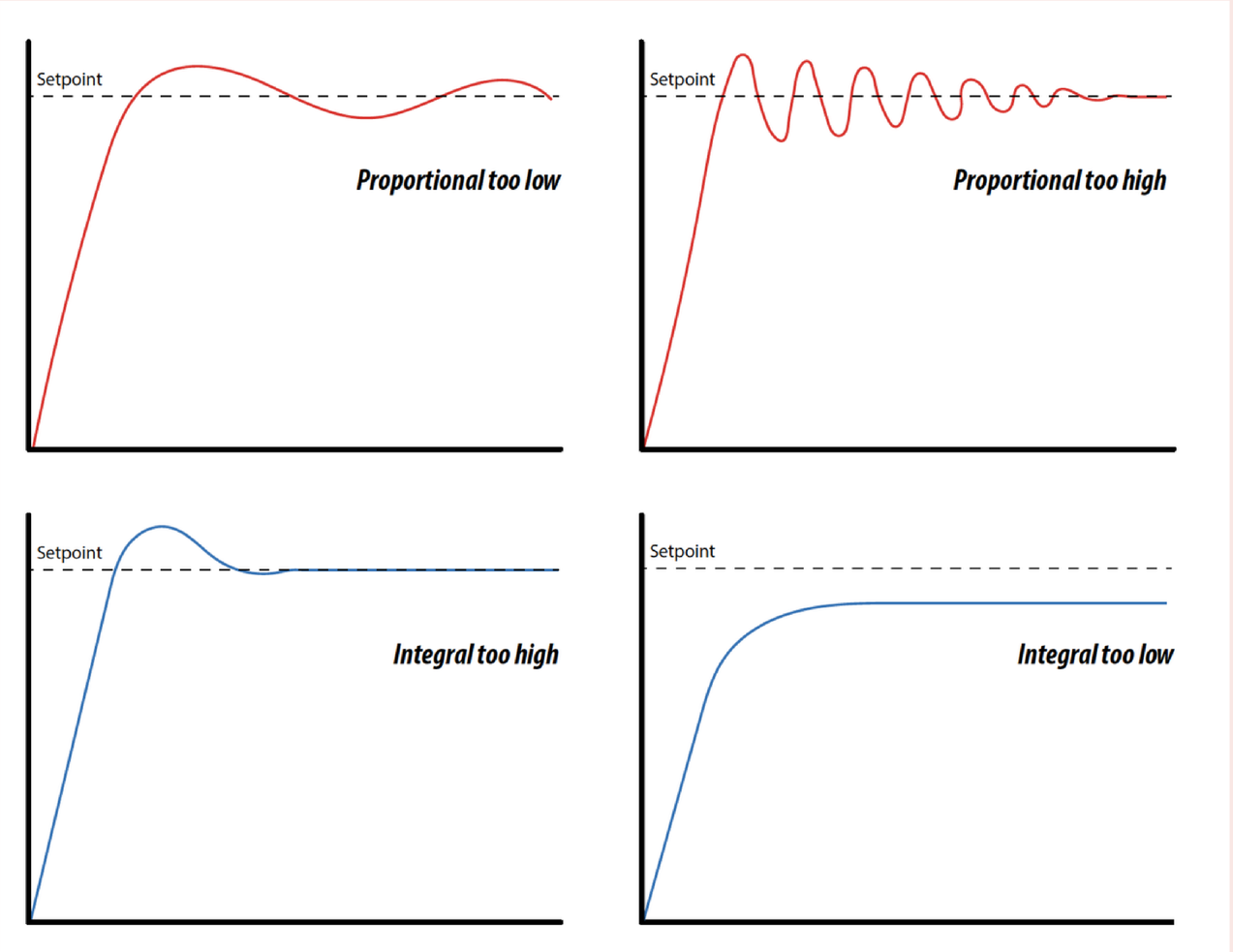


## P- Controller

Proportional or P- controller gives an output that is proportional to current error e (t). It compares the desired or set point with the actual value or feedback process value. The resulting error is multiplied with a proportional constant to get the output. If the error value is zero, then this controller output is zero.

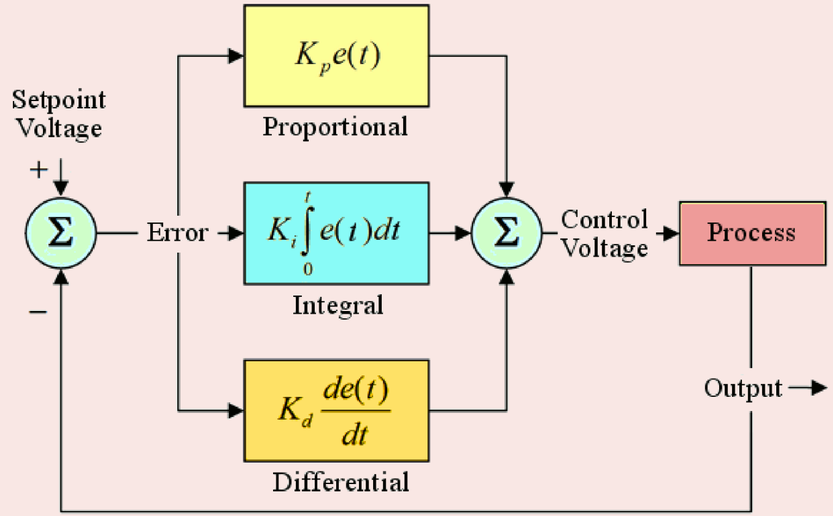
## I-Controller

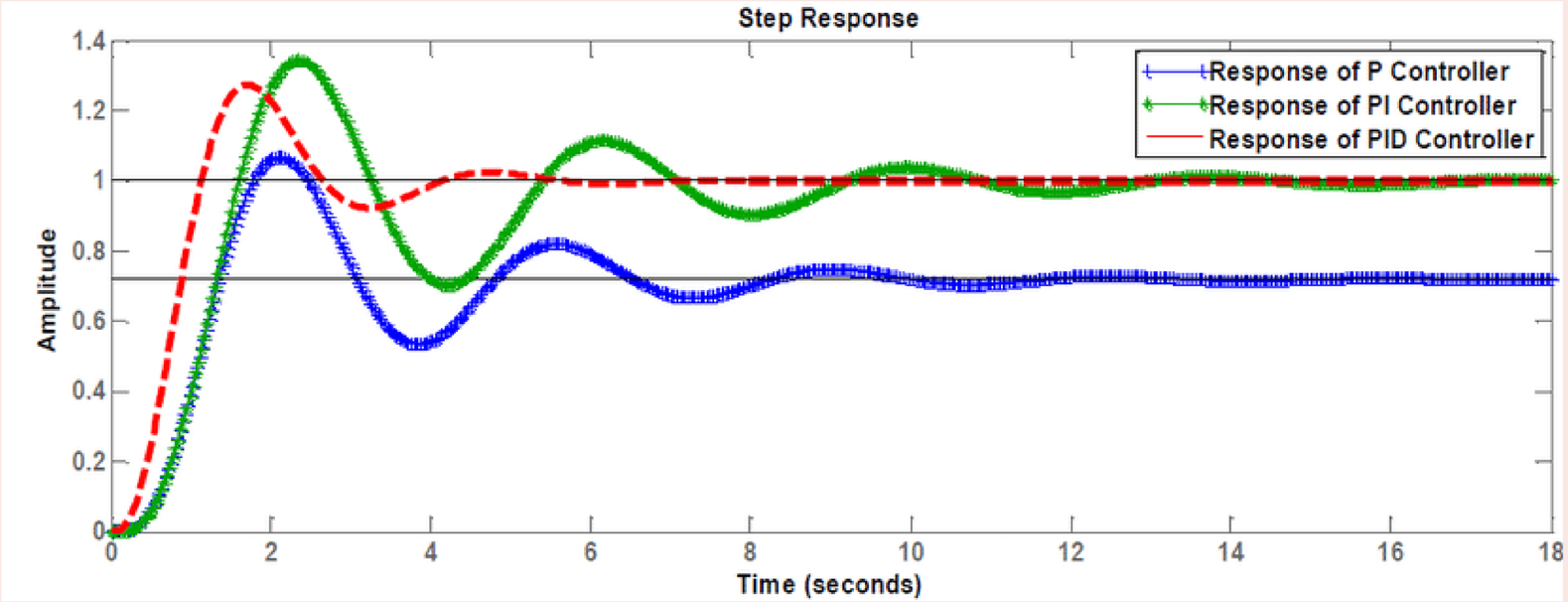
Due to the limitation of p-controller where there always exists an offset between the process variable and setpoint, I-controller is needed, which provides necessary action to eliminate the steady-state error. It integrates the error over a period of time until the error value reaches zero. It holds the value to the final control device at which error becomes zero.

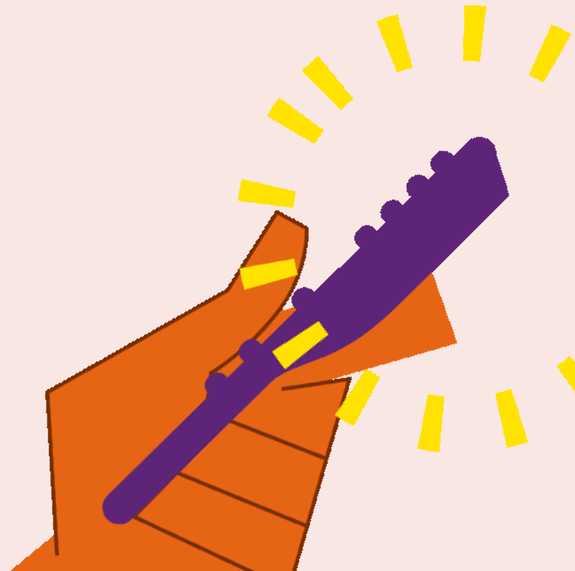


## D-Controller

I-controller doesn’t have the capability to predict the future behavior of error. So it reacts normally once the setpoint is changed. D-controller overcomes this problem by anticipating the future behavior of the error. Its output depends on the rate of change of error with respect to time, multiplied by derivative constant. It gives the kick start for the output thereby increasing system response.




Thank
you