Loading in the Pendulum Model

High-level: Load in pendulum class from double\_pendulum.py

Low level:

Text

Description automatically generated with medium confidence

High-level: Grab state of model (state = configurations + velocities, limits)\

In short, the state keeps track of the position and motion of 3-bodied pendulum

Graphical user interface, text

Description automatically generatedLow level:

Graphical user interface

Description automatically generated with medium confidence

Looking at the Actuators

High-level:

The present ActuationModelDoublePendulum is derived from ActuationModel Class, and only adds an actuator to the first joint. We want to change that.

An actuation is a rotational force (torque) applied by actuators on joints:

(controls, state) 🡪 torques

(2 controls, 4-d state (2 angles + 2 angular velocities)) 🡪 (2 torques to actuators on joints)

­Low-level:

# defining a class derived from actuation model

class ActuationModelDoublePendulum(crocoddyl.ActuationModelAbstract):

    def \_\_init\_\_(self, state, actLink):

##### Dimension of control vector is 1

        crocoddyl.ActuationModelAbstract.\_\_init\_\_(self, state, 1)

        self.nv = state.nv #nv is the number of joints

        self.actLink = actLink # This is link where actuator will be placed(in case 1)

    def calc(self, data, x, u):

        data.tau[:] = data.S \* u

    def calcDiff(self, data, x, u):

        data.dtau\_du[:] = data.S

    def createData(self): # This method actually creates actuation model

        data = ActuationDataDoublePendulum(self)

        return data

class ActuationDataDoublePendulum(crocoddyl.ActuationDataAbstract):

# Inheritance from ActuationDataAbstract

    def \_\_init\_\_(self, model):

        crocoddyl.ActuationDataAbstract.\_\_init\_\_(self, model)

        if model.nu == 1:

            self.S = np.zeros(model.nv)

        else: # What is self.S?

            self.S = np.zeros((model.nv, model.nu))

        if model.actLink == 1:

            self.S[0] = 1

        else:

            self.S[1] = 1

This looks messy, lets try another route.

Looking at the Crocoddyl website, I see that they have another actuation model (ActuationModelFullTpl), which applies input controls for all nv dimensions of the system (in this case nv = 2)