Step 1: Change the actuation Model

From:

ActuationModelDoublePendulum(state, actLink=1)

To:

actuation = crocoddyl.ActuationModelFull(state)

This new model successfully solve optimization problem in before the time horizon:

Chart

Description automatically generated

Step 2: Allow nonzero reference state

Cannot use CostModelDoublePendulum as it does not accept reference as parameter

Instead try to use ActivationModelWeightedQuad outside of the Cost Model Double Pendulum

Invalid argument: nr is equals to 4

Where nr = dimension of cost

Let’s change the activation function

From:

np.array([1.] \* 4 + [0.1] \* 2)

To:

np.array([1.] \* 4)

Testing with reference = [2 0 2 0] this performs poorly, staying far away from reference state for long period of time, and them leaping to stop at end.

Chart, line chart

Description automatically generated

Cost function:

Chart

Description automatically generated with low confidence

What could this mean? It looks like it incurs a high cost in the beginning, and then almost 0 cost over the rest of the time frame

1. We could increase the cost of being away from the reference state (dramatically)

Changing the weight of xReg cost by factor of 10:

runningCostModel.addCost("xGoal", xRegCost, 1e-4 / dt)

At 1/1000 / dt it distributes cost more but still stagnant for most of time horizon

Chart, line chart

Description automatically generated Chart, line chart, scatter chart

Description automatically generated