Step 1: Change the actuation Model

From:

ActuationModelDoublePendulum(state, actLink=1)

To:

actuation = crocoddyl.ActuationModelFull(state)

This new model successfully solves the optimization problem (go to 0,0,0,0) 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 generatedChart, line chart, scatter chart

Description automatically generated

1. What if we added activation function for the controls?

uActivation = crocoddyl.ActivationModelWeightedQuad(np.array([0.1] \* 2))

uRegCost = crocoddyl.CostModelResidual(state, uActivation, uResidual)

No Cigar

Chart, line chart

Description automatically generatedA picture containing chart

Description automatically generated

1. It seems like both controls are spiking towards the end, what if we put a quadratic barrier to prevent this?

lb = np.array(xRef) + np.array([-1, -.5, 1, -.5])

ub = np.array(xRef) + np.array([1, .5, 1, .5])

bounds = crocoddyl.ActivationBounds(lb, ub)

xQuadBar = crocoddyl.ActivationModelQuadraticBarrier(bounds)

xBarCost = crocoddyl.CostModelResidual(state, xQuadBar, xResidual)

xBarCost = crocoddyl.CostModelResidual(state, xQuadBar, xResidual)

runningCostModel.addCost("xBar", xBarCost, 1e-5 / dt)

Chart, line chart

Description automatically generated

Still no improvement. Perhaps if increase this Quadratic barrier cost?

Also, it looks like here we are penalizing angular velocity, what if we increased the range of lb, ub?

* Still no good

What if we try a weighed quadratic instead, that greatly punished u-Residual at each point in time? Perhaps that would incentivize model to go to reference as quickly as possible

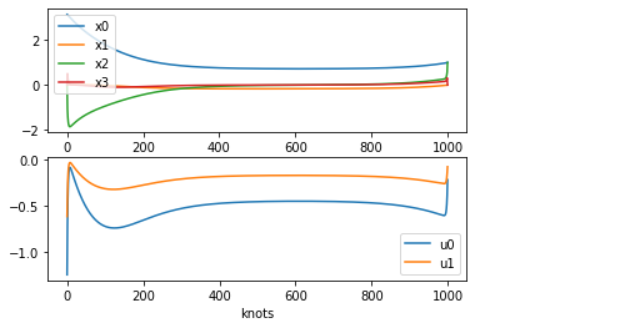
lb = np.array(xRef) + np.array([-.1, -.1, 1, -.1])

ub = np.array(xRef) + np.array([.1, .1, 1, .1])

w = np.array([1, 1, 1, 1])

xQuadBar = crocoddyl.ActivationModelWeightedQuadraticBarrier(bounds, w)

runningCostModel.addCost("xBar", xBarCost, 1e-6 / dt)



This is looking a little bit better, not as strong spike on each control. Now let’s focus on penalizing only the x2

* That didn’t work, it only made these graphs jerkier

At the end we see hug spike in velocity, what if we heavily penalize velocity past 1?

w = np.array([.1, 3, .1, 3])

* No improvement

What if we removed the regular quadratic cost?

# runningCostModel.addCost("xGoal", xRegCost, 1e-5 / dt)

Chart, line chart

Description automatically generated

This looks like slightly smoother movement going from:

state=0, 0, 0, 0 🡪

state= 1, 0, 1, 0

Sharp movements at beginning though

What if additionally punished these spikes in velocity?

w = np.array([1, 2, 1, 2])

Chart, line chart

Description automatically generated

This looks slightly better.

What if we increased lb and ub?

z = .7

lb = np.array(xRef) + np.array([-z, -z, -z, -z])

ub = np.array(xRef) + np.array([z, z, z, z])

Chart, line chart

Description automatically generatedNow x2 gets to 1 much faster. Now how can we get x0 there faster? *Stuck.*