A LQR and MPC Algorithms

Algorithm 1 $\mathrm{LQR}_T(x_{\text{init}}; C, c, F, f)$ *Solves Equation (2) as described in Levine [2017]*

The **state space** is n-dimensional and the **control space** is m-dimensional.
$T \in \mathbb{Z}_+$ is the **horizon length**, the number of nominal timesteps to optimize for in the future.
$x_{\text{init}} \in \mathbb{R}^n$ is the initial state
$C \in \mathbb{R}^{T \times n+m \times n+m}$ and $c \in \mathbb{R}^{T \times n+m}$ are the quadratic cost terms. Every C_t must be PSD.
$F \in \mathbb{R}^{T \times n \times n+m}$ $f \in \mathbb{R}^{T \times n}$ are the affine cost terms.

▷ **Backward Recursion**
$V_T = v_T = 0$
for t = T to 1 **do**
$Q_t = C_t + F_t^\top V_{t+1} F_t$
$q_t = c_t + F_t^\top V_{t+1} f_t + F_t^\top v_{t+1}$
$K_t = -Q_{t,uu}^{-1} Q_{t,ux}$
$k_t = -Q_{t,uu}^{-1} q_{t,u}$
$V_t = Q_{t,xx} + Q_{t,xu} K_t + K_t^\top Q_{t,ux} + K_t^\top Q_{t,uu} K_t$
$v_t = q_{t,x} + Q_{t,xu} k_t + K_t^\top q_{t,u} + K_t^\top Q_{t,uu} k_t$
end for

▷ **Forward Recursion**
$x_1 = x_{\text{init}}$
for t = 1 to T **do**
$u_t = K_t x_t + k_t$
$x_{t+1} = F_t \begin{bmatrix} x_t \\ u_t \end{bmatrix} + f_t$
end for

return $x_{1:T}, u_{1:T}$

Algorithm 2 $\text{MPC}_{T,\underline{u},\overline{u}}(x_{\text{init}}, u_{\text{init}}; C, f)$ *Solves Equation (10) as described in Tassa et al. [2014]*

The **state space** is n-dimensional and the **control space** is m-dimensional.
$T \in \mathbb{Z}_+$ is the **horizon length**, the number of nominal timesteps to optimize for in the future.
$\underline{u}, \overline{u} \in \mathbb{R}^m$ are respectively the control **lower-** and **upper-bounds**.
$x_{\text{init}} \in \mathbb{R}^n, u_{\text{init}} \in \mathbb{R}^{T \times m}$ are respectively the initial state and nominal control sequence
$C : \mathbb{R}^{n \times m} \to \mathbb{R}$ is the non-convex and twice-differentiable **cost function**.
$F : \mathbb{R}^{n \times m} \to \mathbb{R}^n$ is the non-convex and once-differentiable **dynamics function**.

$x_1^1 = x_{\text{init}}$
for t = 1 to T-1 **do**
  $x_{t+1}^1 = f(x_t, u_{\text{init},t})$
end for
$\tau^1 = [x^1, u_{\text{init}}]$

for i = 1 to *[converged]* **do**
  for t = 1 to T **do**
    ▷ Form the *second-order Taylor expansion* of the cost as in Equation (12)
    $C_t^i = \nabla^2_{\tau_t^i} C(\tau_t^i)$
    $c_t^i = \nabla_{\tau_t^i} C(\tau_t^i) - (C_t^i)^\top \tau_t^i$

    ▷ Form the *first-order Taylor expansion* of the dynamics as in Equation (13)
    $F_t^i = \nabla_{\tau_t^i} f(\tau_t^i)$
    $f_t^i = f(\tau_t^i) - F_t^i \tau_t^i$
  end for
  $\tau_{1:T}^{i+1} = \text{MPCstep}_{T,\underline{u},\overline{u}}(x_{\text{init}}, C, f, \tau_{1:T}^i, C^i, c^i, F^i, f^i)$
end for

function $\text{MPCstep}_{T,\underline{u},\overline{u}}(\, x_{\text{init}}, C, f, \tau_{1:T}, \tilde{C}, \tilde{c}, \tilde{F}, \tilde{f})$
  ▷ C, f are the *true cost* and *dynamics* functions. $\tau_{1:T}$ is the *current trajectory* iterate.
  ▷ $\tilde{C}, \tilde{c}, \tilde{F}, \tilde{f}$ are the *approximate cost* and *dynamics* terms around the current trajectory.

  ▷ **Backward Recursion:** Over the linearized trajectory.
  $V_T = v_T = 0$
  for t = T to 1 **do**
    $Q_t = \tilde{C}_t + \tilde{F}_t^\top V_{t+1} \tilde{F}_t$
    $q_t = \tilde{c}_t + \tilde{F}_t^\top V_{t+1} \tilde{f}_t + \tilde{F}_t^\top v_{t+1}$

    $k_t = \text{argmin}_{\delta u} \;\; \frac{1}{2} \delta u^\top Q_{t,uu} \delta u + Q_x^\top \delta u \;\; \text{s.t.} \;\; \underline{u} \leq u + \delta u \leq \overline{u}$
    ▷ Can be solved with a *Projected-Newton method* as described in Tassa et al. [2014].
    ▷ Let f, c respectively index the *free* and *clamped* dimensions of this optimization problem.

    $K_{t,f} = -Q_{t,uu,f}^{-1} Q_{t,ux}$
    $K_{t,c} = 0$

    $V_t = Q_{t,xx} + Q_{t,xu} K_t + K_t^\top Q_{t,ux} + K_t^\top Q_{t,uu} K_t$
    $v_t = q_{t,x} + Q_{t,xu} k_t + K_t^\top q_{t,u} + K_t^\top Q_{t,uu} k_t$
  end for

  ▷ **Forward Recursion and Line Search:** Over the true cost and dynamics.
  repeat
    $\hat{x}_1 = \tau_{x_1}$
    for t = 1 to T **do**
      $\hat{u}_t = \tau_{u_t} + \alpha k_t + K_t(\hat{x}_t - \tau_{x_t})$
      $\hat{x}_{t+1} = f(\hat{x}_t, \hat{u}_t)$
    end for
    $\alpha = \gamma \alpha$
  until $\sum_t C([\hat{x}_t, \hat{u}_t]) \leq \sum_t C(\tau_t)$

  return $\hat{x}_{1:T}, \hat{u}_{1:T}$
end function

B Imitation learning experiment losses

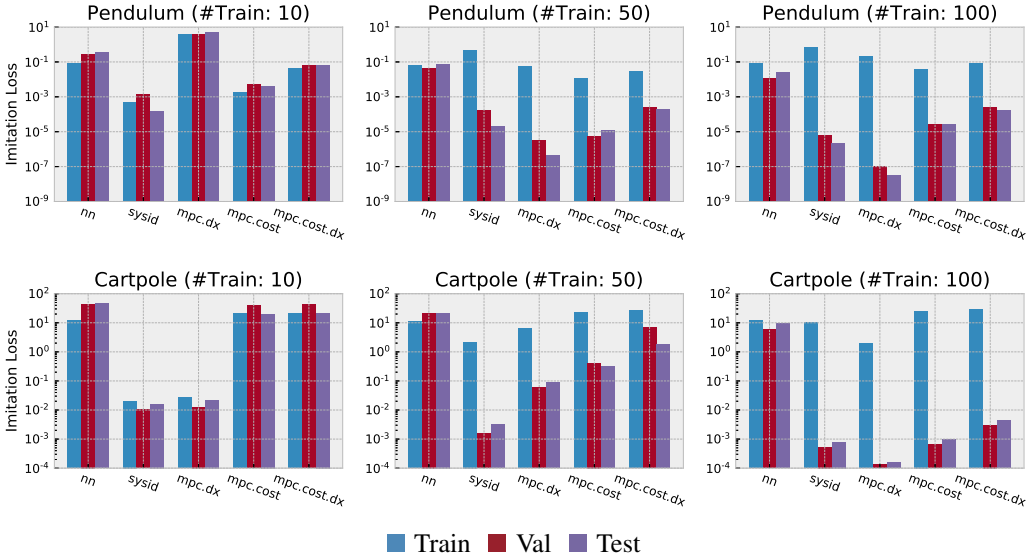

Figure 6: Learning results on the (simple) pendulum and cartpole environments. We select the best validation loss observed during the training run and report the corresponding train and test loss. Every datapoint is averaged over four trials.