Feasibility-prone Differential Dynamic Programming
Is DDP a Multiple Shooting Algorithm?

N. Mansard – LAAS-CNRS

March 27, 2020

1 Introduction

1.1 Problem definition

We are interested to find an approximate solution to the following optimal control problem OCP:

$$\min_{\underline{x},\underline{u}} \int_0^{\mathbb{T}} \ell(x(t),u(t),t)dt + \ell_{\mathbb{T}}(x(\mathbb{T}))$$
$$s.t. \quad x(0) = f_0$$
$$\forall t \in [0,\mathbb{T}], \quad \dot{x}(t) = f(x(t),u(t),t))$$

where $\underline{x} : t \to x(t)$ is the state trajectory, $\underline{u} : t \to u(t)$ is the control trajectory, ℓ is the integral –running– cost, ℓ_T is the terminal cost, f_0 is the initial state value, f is the robot dynamics and T, the time interval, is fixed.

The decision variables are $\underline{x}, \underline{u}$, both of infinite dimension. We approximate this problem using a discrete version of it, by following the so-called direct –discretize first, solve second – approach.

1.2 Discretize first

The time interval $[0,\mathbb{T}]$ is divided into T sub-intervals (evenly distributed or not). In each sub-interval t, the control trajectory $\underline{u}_t$ is constrained to be in the span of a given trajectory finite basis, and we represent the trajectory by its coefficient in the function basis (i.e as a vector of finite dimension). We typically write $\underline{u}_t$ as a polynomial, and it is often taken in practice constant on the interval.

The values of $\underline{x}_t$ on the interval t are obtained by integrating the dynamics from the value x_t at the beginning of the sub-interval. As closed-form integrals of f are often not available, $\underline{x}_t$ is approximated by any numerical integration scheme, e.g. Runge-Kutta-4. We then represent $\underline{x}$ by its values at each interval ends, i.e. as a list of $T+1$ elements.

In summary, the control variable $\underline{u}$ is represented by T basis coefficients of the chosen trajectory basis –which often boils to T constant controls– and $\underline{x}$ is represented by $T+1$ states. In the following, we will often abusively use the same symbols for the true object (e.g. the trajectory) and its representation (e.g. the coefficients of its discretization), in the aim of keeping the notations simple. With this choice, the discretized problem can be written as:

$$\min_{\underline{x},\underline{u}} \sum_{t=0}^{T-1} \ell(x_t,u_t) + \ell_T(x_T)$$
$$s.t. \quad x_0 = f_0$$
$$\forall t = 0..T-1, \quad x_{t+1} = f(x_t,u_t)$$

As announce, both ℓ and f now represent the discretization of their respective objects in the original problem. They both typically depend on time (i.e. ℓ_t, f_t) but we omit this dependency in the notation for readability.

1.3 Solve second

This new problem is now a static optimization problem under constraints, typically nonlinear and non-convex (NLP). We will solve it with a sequential-quadratic-programming (SQP) strategy, i.e. by iteratively solving the linear-quadratic-regulator (LQR) problem obtained by computing the linearization of the dynamics f and the quadratic model of the cost ℓ at the current candidate values of $\underline{x}, \underline{u}$.

We denote the derivatives of f by F_x, F_u, and the gradient and the Hessian of ℓ by (L_x, L_u) and $(L_{xx}, L_{xu}, L_{ux}, L_{uu})$, respectively. When possible, we will omit the time indexes for all these quantities. For the LQR case, due to it is a finite-horizon problem, we can consider without loss of generality that the F and L are constant matrices (for general case, you just need to add the evident indices $_t$ into each quantity). We also denote f_t by the drift of f (i.e. change in x when u is zero), whose role is clear for the LQR and whose role will become clear later for solving the NLP. The LQR is then formulated as:

$$\min_{\underline{\Delta x}, \underline{\Delta u}} \Big(\sum_{t=0}^{T-1} \frac{1}{2} \begin{bmatrix} \Delta x^T, \Delta u^T \end{bmatrix} \begin{bmatrix} L_{xx} & L_{xu} \\ L_{ux} & L_{uu} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta u \end{bmatrix} + \begin{bmatrix} L_x & L_u \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta u \end{bmatrix}$$

$$+ \frac{1}{2} \Delta x_T L_{xx} \Delta x_T + L_x \Delta x_T \Big)$$

$$s.t. \quad \Delta x_0 = f_0$$

$$\forall t = 0 \cdots T-1, \quad \Delta x_{t+1} = F_x \Delta x_t + F_u \Delta u_t + f_t$$

This problem is a quadratic program (under linear equality constraints – QP). Various solutions can be chosen to solve the QP. It is obvious to recall that all of them will lead to the same solution, at least neglecting numerical effects related to noise and numerical stability. We will favor two solutions. For understanding the nature of the problem, we will write the solution to this QP by forming the KKT matrix. For solving it in practice, we will use the Ricatti recursion typical in differential dynamic programming (DDP).

1.4 The Russian way: using the Karush-Kuhn-Tucker matrix

1.4.1 Optimality principle

The Lagrangian of the LQR QP is:

$$\mathcal{L}(\underline{\Delta x}, \underline{\Delta u}, \underline{\lambda}) = \sum_{t=0}^{T-1} \Big(\frac{1}{2} \begin{bmatrix} \Delta x_t^T, \Delta u_t^T \end{bmatrix} \begin{bmatrix} L_{xx} & L_{xu} \\ L_{ux} & L_{uu} \end{bmatrix} \begin{bmatrix} \Delta x_t \\ \Delta u_t \end{bmatrix} + \begin{bmatrix} L_x & L_u \end{bmatrix} \begin{bmatrix} \Delta x_t \\ \Delta u_t \end{bmatrix}$$

$$+ \lambda_t (\Delta x_{t+1} - F_x \Delta x_t - F_u \Delta u_t - f_t) \Big) + \frac{1}{2} \Delta x_T L_{xx} \Delta x_T + L_x \Delta x_T$$

1.4.2 Solving the LQR QP

The optimum of the QP is reached for the zero of the gradient of $\mathcal{L}$ with respect to $\underline{x}$, $\underline{u}$ and $\underline{\lambda}$, i.e. when:

$$\begin{bmatrix}
L_{xx} & & & & L_{xu} & & & & -I & F_x^T & & \\
 & \ddots & & & & \ddots & & & & \ddots & \ddots & \\
 & & L_{xx} & & & & L_{xu} & & & & -I & F_x^T \\
 & & & L_{xx} & & & & L_{xu} & & & & -I \\
L_{ux} & & & & L_{uu} & & & & & F_u^T & & \\
 & \ddots & & & & \ddots & & & & & \ddots & \\
 & & L_{ux} & & & & L_{uu} & & & & & F_u^T \\
-I & & & & & & & & & & & \\
F_x & -I & & & F_u & & & & & & & \\
 & \ddots & \ddots & & & \ddots & & & & & & \\
 & & F_x & -I & & & F_u & & & & &
\end{bmatrix}
\begin{bmatrix}
\Delta x_0 \\ \vdots \\ \Delta x_{T-1} \\ \Delta x_T \\ \Delta u_0 \\ \vdots \\ \Delta u_{T-1} \\ \lambda_0 \\ \lambda_1 \\ \vdots \\ \lambda_{T-1}
\end{bmatrix}
= -
\begin{bmatrix}
L_x \\ \vdots \\ L_x \\ L_x \\ L_u \\ \vdots \\ L_u \\ f_0 \\ f_1 \\ \vdots \\ f_{T-1}
\end{bmatrix}$$

Solving this linear equation (by inverting the full-rank KKT matrix) provides both the optimal state and control trajectories $\underline{\Delta x}$, $\underline{\Delta u}$ and the Lagrange multipliers corresponding to the robot dynamics. These multipliers indeed represent the trajectory of the co-state at the shooting points. Solving the LQR QP by searching the zeros of the Lagrangian indeed corresponds to applying the Pontryagin's Minimum Principle (PMP) on the discretize LQR system.

1.4.3 Intuition of the results

Solving the QP provides at the same time the satisfaction of the constraints (i.e. the resulting $\underline{x}$ is a continuous state trajectory that corresponds to the continuous control trajectory $\underline{u}$, following the underlying linear integrator); and the resulting trajectory pair is of minimal (quadratic) cost.

Indeed, solving the QP can be seen as computing a step to modify an initial-guess $\underline{\Delta\bar{x}}$, $\underline{\Delta\bar{u}}$. Such an observation is somehow trivial, as for a QP, any initial guess will lead to the exact same optimum in a single step. However, understanding this observation is important before going to the more complex SQP case. As any initial guess works the same, we typically consider $\underline{\Delta\bar{x}} = 0$ and $\underline{\Delta\bar{u}} = 0$. Then this initial guess is not feasible, i.e. it does not satisfy the constraints (except in the particular case where $f_0 = \cdots = f_{T-1} = 0$). These f_t can then be seen as the gaps in the state trajectory, i.e. the state trajectory is piece-wise feasible inside each shooting interval, but is discontinuous at each shooting node t with a gap between the previous piece of trajectory t and the next one $t+1$ of f_t (f_0 being a gap with respect to the initial guess state $\Delta\bar{x}_0 = 0$).

Then solving the QP corresponds to both making a step that nullifies the gaps f_t and a step that optimizes the trajectory cost.

1.5 The American way: computing the optimal flow

1.5.1 Optimality principle

DDP is generally not associated with the KKT matrix but with the backward Riccati recursion. We denote the Value function (running cost) at time t by $V_t : \Delta x \to \mathbb{R}$, and it represents the minimal cost we can obtained with the state being Δx at time t. We denote the Hamiltonian of the system (Q-value) by $Q_t : \Delta x, \Delta u \to Q_t(\Delta x, \Delta u) = \ell_t(\Delta x, \Delta u) + V \circ \Delta f(\Delta x, \Delta u)$, where the linear dynamics is denoted by $\Delta f(\Delta x, \Delta u) = F_x \Delta x + F_u \Delta u + f_t$.

As the problem is LQR, the Value and Hamiltonian functions have quadratic form; they can be represented by their gradient and their Hessian. It is important to note that the gradient of a quadratic function is not constant but varies according to the point where it is computed. However, the gradient at any point can be easily computed using the gradient at a given point plus the Hessian times the difference between the two points $\nabla(a) = \nabla(b) + \nabla^2 \cdot (a - b)$. And often, we conveniently compute the gradient at the origin.

1.5.2 Solving the backward recursion

From the Bellman principle, we know that $V_t(\Delta x) = \min_{\Delta u} Q_t(\Delta x, \Delta u)$. A backward recursion can then be set up, starting from the final observation that $V_t = \ell_T$. The backward recursion can be computed along any given guess trajectory $\underline{\Delta\bar{x}}, \underline{\Delta\bar{u}}$, to compute Q_t at each shooting node t from V_{t+1}, and then V_t by optimizing $Q_t(., \Delta u)$. It is important to remember that any trajectory can be chosen, as the problem is LQR, hence the optimal flow V can be equivalently recovered from any trajectory $\underline{\Delta\bar{x}}$. In particular, the trajectory does not have to be optimal, feasible, or even continuous.

When computing backward the optimal flow, we only compute the V values at the shooting times. However, the flow exists at any time and it is implicitly handled by the recursion through the integrated dynamics F_x, F_u. Then care has to be taken for discontinuous trajectories. In such a case the flow V_t would be typically computed at Δx_t, while the Jacobians F_x, F_u are computed at the state reached at the end of interval $t-1$, i.e. $\Delta x^+_{t-1} = F_x \Delta x + F_u \Delta u + f_{t-1}$. In particular, when $\underline{\Delta\bar{x}} = 0$ and $\underline{\Delta\bar{u}} = 0$, then $\Delta x^+_{t-1} = f_{t-1}$. The gradient of V in Δx^+_{t-1} is obtained from the gradient of V at $\Delta x_t = 0$ with:

$$V_x^+ = V_x + V_{xx} f_t$$

and, off course, if $f_t = 0$, then they are both equals.

The backward recursion is then twofold. First it propagates the Q function from $Q = \ell + V \circ \Delta f$:

$$\begin{aligned}
Q_{xx} &= L_{xx} + F_x^T V_{xx} F_x \\
Q_{xu} &= L_{xu} + F_x^T V_{xx} F_u \\
Q_{ux} &= L_{ux} + F_u^T V_{xx} F_x \\
Q_{uu} &= L_{uu} + F_u^T V_{xx} F_u \\
Q_x &= L_x + F_x^T V_x^+ \\
Q_u &= L_u + F_u^T V_x^+
\end{aligned}$$

The Value function is then obtained by solving the minimum of $Q(., \Delta u)$

$$\arg\min_{\Delta u} Q(\Delta x, \Delta u) = -k - K\Delta x$$

with $k = Q_{uu}^{-1} Q_u$ and $K = Q_{uu}^{-1} Q_{ux}$. The Value is then:

$$\begin{aligned}
V_{xx} &= Q_{xx} - Q_{xu} K \\
V_x &= Q_x - Q_{xu} k + V_{xx} f
\end{aligned}$$

where the gradient V_x is computed at the end of the previous interval x^+ and not at the shooting state x_t.

To obtain the complete solution $\underline{\Delta x}, \underline{\Delta u}$, a forward pass must then be performed. We discuss it later.

1.5.3 Intuition of the result

While the KKT approach computes the solution using the dual λ variable, the DDP approach computes it using the V, Q auxiliary variables. λ presents the co-state trajectory, while V is the value functions. Both are connected, as the PMP principle writes the optimal control in term of the co-state, while HJB express the optimal policy in term of the Value space gradient.

The optimal flow is evaluated along the candidate trajectory $\underline{\Delta\bar{x}}, \underline{\Delta\bar{u}}$. As the problem is LQR, any initial guess produces the same backward pass, the same V and the same solution. For this reason, choosing $\underline{\Delta\bar{x}} = 0, \underline{\Delta\bar{u}} = 0$ is very relevant. In that case, if f_t is nonzero, then the initial guess is not feasible and the term $V_{xx} f$ in the Value gradient back-propagation is important. This term is a direct application of LQR equation with drift, but it is rarely mentioned in DDP works [**?** **?**]. Its role will become very important in the following.

The solution of the LQR has once more two effects: it generates a feasible trajectory where the gap between x^+_{t-1} and x_t is zero; and this trajectory is optimal.

1.6 Equivalence

Both solutions are equivalent.

Proof: the KKT solution is obtained by applying PMP on the LQR system. The Riccati solution comes from integrating HJB equation on the LQR. In the LQR case, both PMP and HJB are sufficient optimality conditions. Then both solutions are equal.

1.7 Partial step

In many algorithm, the QP step is only partly integrated, i.e. the initial guess is only modified by $\alpha\Delta$ where $\alpha \in [0, 1[$ and $\Delta = \Delta\underline{\Delta x}, \Delta\underline{\Delta u}$ the solution to the QP (for example if considering inequality constraints inside an active-set algorithm, or using the QP as the inner solver inside a SQP). What is the effect of taking a partial step?

For the KKT formulation, the effect is pretty clear. As $\Delta\underline{\Delta x}$ was a step from a zero initial guess $\underline{\Delta \bar{x}} = 0$, making a partial step $\alpha\Delta$ will bridge only a part of the gap. Denoting by $\underline{\Delta x}^\alpha$ the solution obtained after making a step of length $\alpha < 1$, we have:

$$\Delta x_0^\alpha = \alpha f_0$$

$$\Delta x_{t+1}^\alpha - F_x \Delta x_t^\alpha - Fu \Delta u_t^\alpha = \alpha f_t$$

The new solution $\underline{\Delta x}^\alpha, \underline{\Delta u}^\alpha$ is then infeasible and the gaps at each shooting points have only be reduced of $(1 - \alpha)$.

For the DDP formulation, this is less clear as it is not described in the literature. As we started to explain, the complete solution should be obtained by rolling out the quadratic policy k, K from the initial state f_0. But this is only when making a full step. When making a partial step, the KKT partial solution can be obtained if (i) applying only a part of k and (ii) keeping a part of the gap at each shooting node.

1.8 Solving the DDP forward pass with a partial step

The forward pass for a partial step $\alpha \leq 1$ then becomes:

$$\Delta x_0 = \alpha f_0$$

$$\Delta u_t = -\alpha k_t - K_t \Delta x_t$$

$$\Delta x_{t+1} = F_x \Delta x_t + F_u \Delta u_t + \alpha f_t$$

Proof: by recurrence. We denote the $\Delta x^*, \Delta u^*$ the optimal solution given by the KKT and by the DDP for a full step. We show the the proposed forward pass produced the partial KKT step.

$$\Delta x_0 = \alpha f_0 = \alpha \Delta x_0^*$$

Now, assuming the $\Delta x_t = \alpha \Delta x_t^*$, we have:

$$\forall t = 0..T\text{–}1, \qquad \Delta u_t = -\alpha k_t - K_t \alpha \Delta x_t^* = \alpha \Delta u_t^*$$
$$\Delta x_{t+1} = F_x \alpha \Delta x_t^* + F_u \alpha \Delta u_t + \alpha f_t = \alpha \Delta x_{t+1}$$

□

2 A feasibility-prone OCP solver using DDP

So far we have detailed a method to solve a LQR program. Let's now look at the more generic case where cost and dynamics are any smooth functions. The transcription of the OCP is a NLP with nonlinear equality constraints representing the robot dynamics. We solve it with a SQP approach, i.e. at each iteration we compute the LQR corresponding to the tangent (differential) of the OCP at the current candidates of the decision variables; we solve the SQP and obtain a descent direction; and we search along the descent direction for a step of adequate length.

2.1 LQR and descent direction

The LQR is uniquely defined by the gradients of f and ℓ and the Hessian of ℓ at the current guess $\underline{x}, \underline{u}$. The solution of the LQR is also uniquely defined and can be computed by any adequate method, in particular by inverting the KKT or by solving the backward-forward Riccati recursions (at least if not considering the numerical and complexity issues). Both methods will give exactly the same descent directions (neglecting the rounding errors).

2.2 Line search and integration

Once the direction has been computed, any line search algorithm can be implemented. Basically, the idea is to try several directions and to take the longer step which gives a reward that corresponds to what the LQR model predicts. When considering a SQP, two contradictory objectives have to be considered: (i) the cost should decrease similarly to what the quadratic model predicts and (ii) the constraints residual should not increase. The trade-off between these two objectives is typically decided following a merit function, chosen by the user.

It is important to better understand why the constraint residual may increase. First, the current guess may, or may not, be feasible, i.e the state at the end of each shooting interval may, or may not correspond to the value of the state at the beginning of the next interval. Following the names chosen in the LQR case, we name gap the discontinuity at the shooting nodes: the current guess is feasible if and only if all the T gaps are zero.

If all the gaps are zero, the descent direction may make them nonzero because the descent direction is only computed from a linear model of the dynamics F_x, F_u. The longer the step, the more incorrect the linear model, and the larger the gaps will grow. The merit function then adjust the step length to all some gap growth (it is impossible with the linear model to prevent some growth) but forbid to large gaps to appear.

If some gaps are nonzero, then the corresponding f_t in the LQR corresponds to these gaps. In that case, the LQR direction will bridge the gap thanks to the linear prediction f_t, but it will simultaneously increase the gap because the linear prediction is inaccurate. More precisely, the gap at time t after a step $\Delta x, \Delta u$ will be:

$$f(x_t + \Delta x_t, u_t + \Delta u_t) - (x_{t+1} + \Delta x_{t+1}) = f(x_t, u_t) - x_{t+1} + F_x \Delta x_t + F_u \Delta u_t - \Delta x_{t+1} + \circ(\alpha^2)$$

where $\Delta x_{t+1} - F_x \Delta x_t - F_u \Delta u_t = \alpha f_t$ by construction of the LQR, and the step length α is the magnitude of the step $\underline{\Delta x}, \underline{\Delta u}$ that leads to quadratic $\circ(\alpha^2)$ errors of the linear model.

$$f(x_t + \Delta x_t, u_t + \Delta u_t) - (x_{t+1} + \Delta x_{t+1}) = (1 - \alpha)(f(x_t, u_t) - x_{t+1}) + \circ(\alpha^2)$$

The gap evolution is composed of two terms: the first one decreases when α growth, and collapses for full step $\alpha = 1$; the second one growths with α, and vanishes with α small. Only the second one exists when the gap of the current guess is null.

2.3 Nonlinear roll-out

Once more, it is important to recall that the descent direction is the same for KKT and DDP. However, the DDP is typically associated with a particular line search variant. First, recall that the exact classical line search can be implemented with the DDP: for that, the roll-out should be performed on the linear model, and a merit function should be considered.

Yet, the DDP classically observes that a feasible solution is directly obtained by integrating the nonlinear dynamics around a candidate solution $\underline{u}$ from the initial state x_0. As the nonlinear dynamics $f(x, u)$ is not exactly the same as the linear dynamics F_x, F_u, the feedback term K must be used during the integration to avoid the divergence. With such a roll-out, the resulting candidate decision variable is a feasible trajectory, where all the gaps are zero.

This behavior is very different from the one observed with classical line search. We rather suggest that the same gaps than for the linear line search should be used for the nonlinear roll-out. We then impose the gaps at the next candidate solution to be:

$$f(x_t + \Delta x_t, u_t + \Delta u_t) - (x_{t+1} + \Delta x_{t+1}) = (1 - \alpha)(f(x_t, u_t) - x_{t+1})$$

No need here to consider the second-order disturbance term $\circ(\alpha^2)$ as we are considering the exact nonlinear dynamics.

2.4 Discussion

2.4.1 Bridging the gap

It has been observed in multiple shooting that keeping the gaps closed during the entire search might be counterproductive as it makes the NLP search very sensitive to the instabilities of the dynamics. We agree with this observation, which it is correlated to the fact that the DDP tends to be an algorithm with poor exploration (globalization) capabilities. We have demonstrated in our experiment that the DDP is much more prone to face feasibility problem, and discover good solutions despite poor initial guesses, when the gaps are kept during the first part of the search.

2.4.2 Using the true dynamics or its approximation

Using the nonlinear dynamics has some advantages. First, despite intuition, the nonlinear dynamics might be faster to compute, as in robotics F_x and F_u are large matrices with little structure (sparsity) while very efficient nonlinear routines exists to compute $f(x, u)$. On the other hand, taking a linear step provides an exact Newton step with strong convergence guarantees, at least when the NLP is convex (and often this is not the case).

We claim that the choice should be taken by considering the effect on the gaps. With the nonlinear step, the gaps strictly decreases with nonzero step. With the linear step, it does not strictly decreases and we have to relying on the merit function to accept reasonable growth. While we agree that maintaining the gaps open during the search is interesting, and that it might even be interesting to enlarge them for globalization purpose, it is doubtful that the $\circ(\alpha^2)$ term might be an interesting growth direction. Indeed, this term corresponds to some growth direction that are not predicted in the linear model. The LQR is then not informed to choose an interesting value for that perturbation.

We have tried to experiment gap growth coming from the linear prediction error versus other perturbation terms, in particular random, and of course zero (with the nonlinear roll-out). While it seems clear that the term $(1 - \alpha)(f - x)$ is interesting, the interest or noxiousness of $\circ(\alpha^2)$ has not been observed.

In conclusion, we advice to take a nonlinear step while maintaining the gaps open. If it is desirable to effectively relax the continuity constraints, we then advice to really relax the dynamic constraint (e.g. putting it in the cost as a penalty) and not to rely on the unpredicted disturbance $\circ(\alpha^2)$.

3 Conclusion

We have proposed two modifications to the classical DDP algorithm to solve OCP written as NLP problems. First, we modified the backward pass to accept infeasible trajectories, i.e. trajectories were a discontinuity exists at each shooting interval. Second, we modified the line search algorithm to avoid bridging the gap after the first step is taken.

In consequence, the DDP algorithm accepts any initial guess and has a much better globalization capability. We can observe that the behavior is nearly the same as a multiple-shooting solver. Indeed, it is exactly the same if the classical line search is implemented. We did not demonstrate that the feasibility-prone DDP or the multiple shooting is better: the performance

where equivalent on all the problems we have considered. We discussed that the DDP might be easier to implement and make efficient, and we advice to choose it.

4 Expectation model of the FDDP

At each Newton step of a NLP solver, a line-search is performed. One of the simple conditions of acceptance of the step is that the actual improvement of the cost should be similar to the expected improvement provided by the quadratic model. For example, if considering a function f and its model m with $f(x + \Delta x) = f(x) + m(\delta x) + o(\Delta x)$, then a step δx would be accepted if $f(x + \Delta x) - f(x)$ and $m(x)$ are close enough (in practice, the ratio between the two quantities is close to 1).

When considering the DDP solver, computing the expected improvement is more difficult, as the $\underline{x}, \underline{u}$ of the LQR are never explicitly computed: the backward pass only provides k, from which the actual $\underline{x}, \underline{u}$ are obtained using a nonlinear rollout. This section provides efficient evaluation of the expectation model.

4.1 Expectation model in $\underline{x}, \underline{u}$

Let's first write the expectation model in term of the increments $\underline{x}, \underline{u}$ (let's recall that, to keep the notations concise, we use x and u for the LQR variables, while they should be intepreted as "deltas" in the nonlinear optimization algorithm). If making a step of length α (typically in $]0, 1]$) in the direction $\underline{x}, \underline{u}$, then the improvement of the cost should have the following form:

$$\Delta = \Delta_1 \alpha + \frac{1}{2} \Delta_2 \alpha^2$$

Δ_1 is the sum at each shooting node of the cost gradient times the change in x and u:

$$\Delta_1 = \sum_{t=0}^{T} L_{xt}^T x_t + L_{ut}^T u_t \tag{1}$$

(to keep the sum simpler, we treat T similarly to the other nodes, by introducing $L_{uT} = 0$).

4.1.1 Linear rollout

The states and controls are obtained from a linear roll-out as:

$$x_{t+1} = F_{xt} x_t + F_{ut} u_t + f_{t+1}$$

$$u_t = K_t x_t + k_t$$

Propagating these two equations, we get:

$$x_{t+1} = (F_{xt} + F_{ut} K_t) x_t + F_{ut} k_t + f_{t+1} = F_{t-1} x_{t-1} + c_{t+1}$$

with $F_t = F_{xt} + F_{ut} K_t$ and $c_{t+1} = F_{ut} k_t + f_{t+1}$ (with $c_0 = f_0$). And finally:

$$x_t = F_{t-1} ... F_0 c_0 + F_{t-1} ... F_1 c_1 + ... + F_{t-1} c_{t-1} + c_t \tag{2}$$

$$= \sum_{i=0}^{t} F_{t-1} ... F_i c_i \tag{3}$$

4.1.2 First-order model Δ_1

Replacing u_t by $k_t + K_t x_t$, the first-order term is:

$$\Delta_1 = \sum_{t=0}^{T} (L_{xt} + K_t^T L_{ut})^T x_t + \sum_{t=0}^{T} L_{ut}^T k_t \tag{4}$$

where we denote $l_t = L_{xt} + K_t^T L_{ut}$ to simplify the notation. Putting (**??**) in (**??**), we get:

$$\Delta_1 = \sum_{t=0}^{T} l_t \sum_{i=0}^{t} F_{t-1} ... F_i c_i + L_{ut}^T k_t \tag{5}$$

$$= \sum_{i=0}^{T} c_i^T \sum_{t=i}^{T} F_t^T ... F_T^T l_t + k_i^T L_{ui} \tag{6}$$

Each term of the sum is composed of a product of f_i and a product of k_i, and can then be evaluated from the result of the backward pass. Let's exhibit these 2 terms. The term in f_i is:

$$\Delta_{ft} = F_i^T ... F_T^T l_i = L_{xi} + F_{xi}^T \Delta_{fi+1} + K_i^T (L_{ui} + F_{ui} \Delta_{fi+1})$$

The term in k_i is:

$$\Delta_{ft} = l_{ui} + F_{ui}^T ... F_T^T l_i = L_{ui} + F_{ui}^T \Delta_{fi+1}$$

In the case f_i are all zeros, we can recognize that Δ_f is the value gradient and Δ_k is the Hamiltonian control gradient: $\Delta_f = V_x$ and $\Delta_k = Q_u$. In that case, we simply have:

$$\Delta_1 = \sum_{t=0}^{T} Q_{ut}^T k_t$$

In the general case where the LQR is not drift-free, then Δ_f and Δ_k must be collected during the backward pass while propagating V_x and Q_u. The cost is similar, and an order of magnitude less than propagating the Value Hessians.

4.1.3 Second-order term Δ_2

This section is empty, work remains to be done here.

4.1.4 The simple case where $T = 1$

It is disapointing that the expectation model is so simple in the drift-free case and only depends on backward-computed quantities, while it is so complex and requires to compute additional quantities in the general case. Let's investigate that. The intuition is that the expectation model should only depends on the gradient and hessians of the Value and Hamiltonian functions.

In the case where we only consider one control u_0, the expectation model is:

$$\begin{aligned}
\Delta_1 &= L_{x0}^T x_0 + L_{u0}^T u_0 + L_{x1}^T x1 \\
&= L_0^T f_0 + L_{u0} k_0 + L_{x1} F_0 f_0 + L_{x1} F_{u0} k_0 + L_{x1} f_1 \\
&= (L_0 + F_0^T L_{x1})^T f_0 + (L_{u0} + F_{u0}^T L_{x1})^T k_0 + L_{x1} f_1
\end{aligned}$$

We nearly recognize the gradients V_{x0}, Q_{u0}, V_{x1} respectively in factor of f_0, k_0, f_1, but some terms are missing:

$$V_{x0} = L_0 + F_0^T (L_{x1} + L_{xx1} f_1) + L_{xx0} f_0$$

$$Q_{u0} = L_{u0} + F_{u0}^T (L_{x1} + L_{xx1} f_1)$$

$$V_{x1} = L_{x1} + L_{xx1} f_1$$

Basically, the missing terms correspond to the re-linearization of the gradient at the f_t points at the end of the intervals. Then, we get:

$$\begin{aligned}\Delta_1 &= V_{x0}^T f_0 + Q_{u0}^T k_0 + V_{x1}^T f_1 - \big(f_0^T V_{xx0} f_0 + f_0^T F_0^T L_{xx1} f_1 + k_0^T F_{u0} L_{xx1} f_1 + f_1^T V_{xx1} f_1\big) \\ &= V_{x0}^T f_0 + Q_{u0}^T k_0 + V_{x1}^T f_1 - \big(f_0^T V_{xx0} x_0 + f_1^T V_{xx1} x_1\big)\end{aligned}$$

The second-order term is:

$$\begin{aligned}\Delta_2 &= f_0^T V_{xx0} f_0 + k_0^T Q_{uu0} k_0 + f_1^T V_{xx1} f_1 + 2(f_0^T F_0^T L_{xx1} f_1 + k_0^T F_{u0} L_{xx1} f_1) \\ &= f_0^T V_{xx0} f_0 + k_0^T Q_{uu0} k_0 + f_1^T V_{xx1} f_1 + 2\big(f_1^T V_{xx1}(x_1 - f_1)\big) \\ &= -f_0^T V_{xx0} f_0 + k_0^T Q_{uu0} k_0 - f_1^T V_{xx1} f_1 + 2\big(f_0^T V_{xx0} x_0 + f_1^T V_{xx1} x_1\big)\end{aligned}$$

We can recognize in the additional terms (the 2 last ones) the same terms as in Δ_1. Nicely, they will cancel out in the case we make a full step $\alpha = 1$:

$$\Delta(\alpha) = \alpha(\Delta_1 + \frac{\alpha}{2}\Delta_2)$$

$$\Delta(1) = V_{x0}^T f_0 + Q_{u0}^T k_0 + V_{x1}^T f_1 - \frac{1}{2} f_0^T V_{xx0} f_0 + \frac{1}{2} k_0^T Q_{uu0}^T k_0 - \frac{1}{2} f_1^T V_{xx1} f_1$$

But they do not cancel out in the general case:

$$\begin{aligned}\Delta(\alpha) = \alpha\Big(V_{x0}^T f_0 + Q_{u0}^T k_0 + V_{x1}^T f_1 + \frac{\alpha}{2}(-f_0^T V_{xx0} f_0 - f_1^T V_{xx1} f_1 + k_0^T Q_{uu0}^T k_0) \\ +(\alpha - 1)(f_0^T V_{xx0} x_0 + f_1^T V_{xx1} x_1)\Big)\end{aligned}$$

4.2 Extending to $T > 1$ by recurence

We can now work by recurence to extend the exact same shape to $T > 1$.

On the first order term, addind a new time step will add two terms in k_1 and f_2 where respectively L_{x2} and $(L_{u1} + F_{u1}^T L_{x2})$ are in factor, and also extends the previous factors. The new factors have the same form as the previous ones and can be handled similarly. The extension of the previous factors simply corresponds to the extension of the preview horizon when writing V_x and Q_u. As previously, we are missing the $L_{xx} f$ terms (corresponding to the relinearization), that can be collected. Each of this additional term is a product term involving two f or one f and one k. Regrouping them by decreasing order of the f index, this finally boils to the sum of the $f^T V_{xx} x$:

$$\Delta_1 = \sum_{t=0}^{T} V_{xt}^T f_t + Q_{ut}^T k_t - f_t^T V_{xxt} x_t$$

(with again the simplification of treating symmetrically the last time step with $k_T = 0$).

Similar observations can be made on the second-order term, and lead to:

$$\Delta_2 = \sum_{t=0}^{T} k_t^T Q_{uut}^T k_t - f_t^T V_{xxt} f_t + 2 f_t^T V_{xxt} x_t$$

The expectation model is finally:

$$\Delta(\alpha) = \alpha \sum_{t=0}^{T} V_{xt}^T f_t + Q_{ut}^T k_t + \frac{\alpha}{2}\Big(k_t^T Q_{uut}^T k_t - f_t^T V_{xxt} f_t\Big) + (\alpha - 1) f_t^T V_{xxt} x_t$$

4.3 Line-search algorithm

First, let us note that if all the gaps f_t are null, it is simply:

$$\begin{aligned}\Delta(\alpha) &= \alpha\big(\sum Q_u^T k + \frac{\alpha}{2} k^T Q_{uu} k\big) \\ &= \alpha(\frac{\alpha}{2} - 1) \sum\ Q_u^T\ Q_{uu}^{-1}\ Q_u\end{aligned}$$

This is always negative.

4.3.1 Merit function ... or not

However, Δ can be positive (i.e. corresponds to an increase of the cost function) when some gap f_t are nonzero. This corresponds to the expected behavior of an SQP algorithm: a step is used to reduce the error in the constraints, which can makes the cost function increases. The point is to monitor both the decrease or the increase of the cost function when reducing the gaps in the trajectory. One objective is to find a line-search strategy that holds (at least is consistent) whether the gaps are nonzero or zero. Let us first consider the case where some of the gaps are nonzero.

We introduce the following merit function:

$$\phi(\underline{x}, \underline{u}) = \ell(\underline{x}, \underline{u}) + \mu \sum_{t=0}^{T} \|c_t(\underline{x}, \underline{u})\|_1$$

where ℓ is the total cost function (integral plus terminal) and the constraints c_t are:

$$c_{t+1} = x_{t+1} - f(x_t, u_t) = f_{f+1}$$

$$c_0 = x_0 - x_0^* = f_0$$

where the f_t have already been introduced as the trajectory gaps (defects). We consider how ϕ changes when changing $\underline{x}, \underline{u}$ in the direction $\underline{\Delta x}, \underline{\Delta u}$:

$$\underline{x}' = \underline{x} + \alpha \underline{\Delta x}$$

$$\underline{u}' = \underline{u} + \alpha \underline{\Delta u}$$

We abusively denote by ϕ the merit changes along the line search:

$$\phi(\alpha) := \phi(\underline{x} + \alpha\underline{\Delta x}, \underline{u} + \alpha\underline{\Delta u}) - \phi(\underline{x}, \underline{u})$$

We have:

$$\phi(\alpha) = \ell' - \ell - \alpha\mu \sum_{t=0}^{T} \|f_t\|_1$$

As the f_t does not depend on α (thanks to the nonlinear rollout used in the forward pass, as explained in the first part of this document), we can always find a penalization μ that makes this function decreases. This means that, if μ is large enough, any step from an infeasible guess would be accepted. The drawback is that the step might induces very large increase of the cost function. In particular, the cost increase might be much larger than predicted by the LQR model, in particular when the initial guess is far from being feasible (for unstable dynamics system, when the initial control guess is very far from stabilizing the initial state trajectory).

As we now that $\sum \|f_t\|_1$ will decrease with nonzero α, we rather suggest to only consider the first term $\ell' - \ell$. This term exactly corresponds to the expectation model that we described above.

$$\ell' - \ell = \Delta(\alpha) + \mathcal{O}(\alpha^3)$$

4.3.2 Goldstein condition

We cannot use it a second order version of the Wolfe (Armijo) conditions, first because Δ might be positive (not a descent direction), and second because strong Wolfe conditions uses the gradient at the next candidate point, which are very expensive to compute in our case. We rather suggest to take a second-order version of the Goldstein conditions, i.e. accept a step if the actual cost change is similar to the expected cost change:

$$b_1 \leq \frac{\ell' - \ell}{\Delta(\alpha)} \leq b_2$$

with b_1, b_2 are two adjustable parameters. More precisely, if Δ is negative (the direction is descending), this imposes that:

$$\ell' - \ell \leq b_1 \Delta(\alpha)$$

i.e. that the cost decrease at least a fraction of the expectation model. A contrario, if Δ is positive (the direction is ascending), this imposes that:

$$\ell' - \ell \leq b_2 \Delta(\alpha)$$

i.e. the cost does not increase more than a multiple of the expectation model. In practice, we suggest to use $b_1 = 0.1$ and $b_2 = 2$. This might be better replaced by a switch to avoid the quotient. The condition finally is to accept the step if:

$$\ell' - \ell \leq \begin{cases} b_1 \Delta(\alpha) & \text{if } \Delta(\alpha) \leq 0 \\ b_2 \Delta(\alpha) & \text{otherwise} \end{cases}$$

4.3.3 Approximating Δ with a nonlinear rollout

The expectation model exhibited above implies the explicit values of the changes in the state trajectory in the term $f_t^T V_{xx} \Delta x_t$. However, the DDP algorithm never explicitly computes the Δx_t, but rather directly computes the next x'_t in the rollout, using the nonlinear dynamics. We do not have the linear direction Δx_t, however, we can easily computes the change in the state trajectory by $x'_t(\alpha) - x_t$ where $x'_t(\alpha)$ is the state reached at time t when applying the changes in the control trajectory and at the trajectory gaps. We then set:

$$\Delta x_t = x'_t - x_t$$

And we modify the expectation model accordingly:

$$\Delta(\alpha) = \alpha \sum_{t=0}^{T} V_{xt}^T f_t + Q_{ut}^T k_t + \frac{\alpha}{2}\left(k_t^T Q_{uut}^T k_t - f_t^T V_{xxt} f_t\right) + (\alpha - 1) f_t^T V_{xxt}(x'_t(\alpha) - x_t)$$

Again, this boils down to the sum of the $Q.k$ when the gaps are all zero.