https://github.com/nmansard/wan24

Optimal control for walking robots

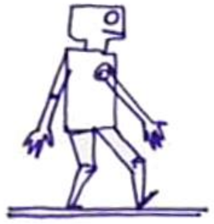
Theory and practice with Crocoddyl

Nicolas Mansard

Gepetto, LAAS-CNRS & ANITI

#04: Differential Dynamic Programming

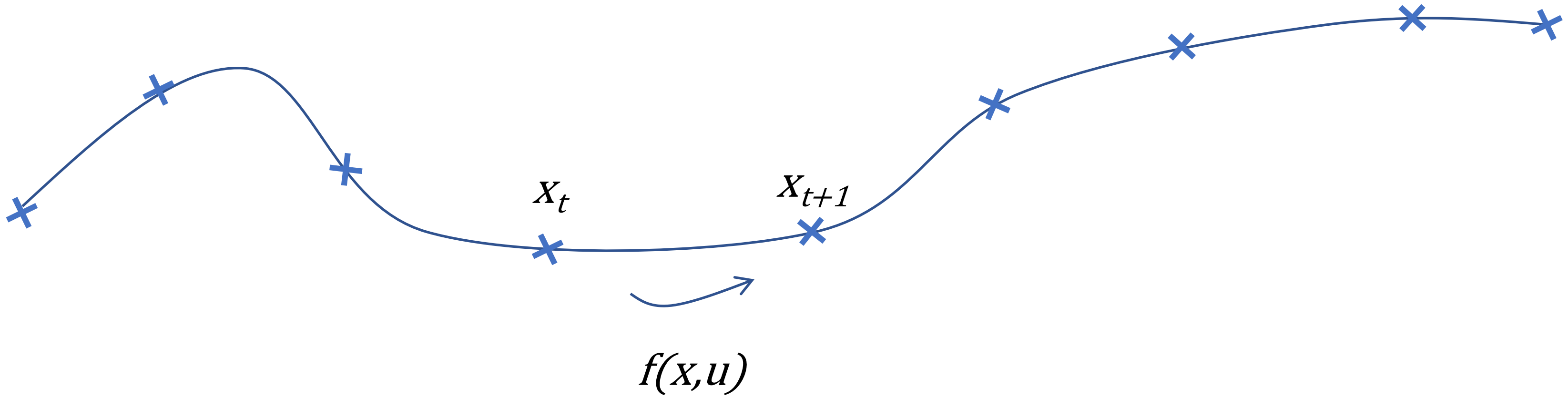




Multiple views on DDP

1. DDP as iterative LQR

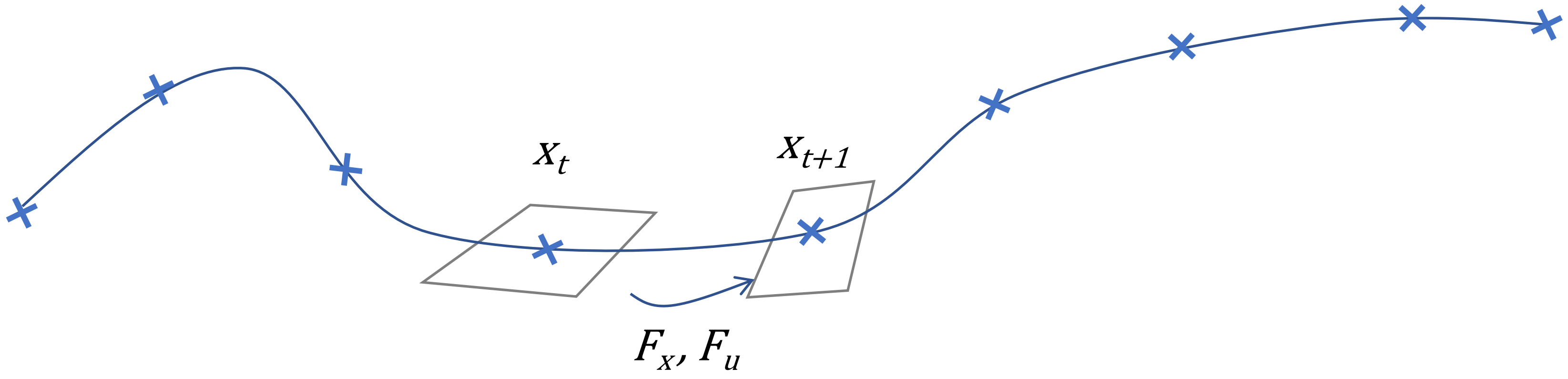
DDP as iterative LQR



- "Next-step" is a nonlinear function

$$\Delta x' = f(x+\Delta x,\ u+\Delta u) - f(x,\ u)$$

DDP as iterative LQR



- "Next-step" is a nonlinear function

$$\Delta x' = f(x+\Delta x,\, u+\Delta u) - f(x, u)$$

- Approximate by

$$\Delta x' = f(x, u) + F_x \Delta x + F_u \Delta u - f(x, u)$$

DDP as iterative LQR

- Nonlinear optimal control problem

$$\min_{\{x\},\{u\}} \sum_{t=0}^{T-1} l(x_t, u_t) \; + l_T(x_T)$$

$$\text{s.t.} \quad \forall \text{t=0..T-1} \quad x_{t+1} = f(x_t, u_t)$$

- Linear-Quadradic problem … solved with Ricatti recursion (textbook)

$$\min_{\{\Delta x\},\{\Delta u\}} \sum_{t=0}^{T-1} \begin{pmatrix} L_x \\ L_u \end{pmatrix}^T \begin{pmatrix} \Delta x_t \\ \Delta u_t \end{pmatrix} + \frac{1}{2} \begin{pmatrix} \Delta x_t \\ \Delta u_t \end{pmatrix}^T \begin{pmatrix} L_{xx} & L_{xu} \\ L_{ux} & L_{uu} \end{pmatrix} \begin{pmatrix} \Delta x_t \\ \Delta u_t \end{pmatrix} + \cdots$$

$$\text{s.t.} \quad \forall \text{t=0..T-1} \; \Delta x_{t+1} = F_x \Delta x_t + F_u \Delta u_t$$

DDP as iterative LQR

- Algorithm iLQR

```
Initialize with a given trajectory {x0},{u0}
Repeat
    Linearize/Quadratize the OCP
    Compute the LQR policy
    Simulate (roll-out) with LQR regulator
Until local minimum is reached
```

Multiple views on DDP

2. DDP as a 2-pass algorithm

DDP as a 2-pass algorithm

$$V_t = \min_{u_t} l(x_t, u_t) + V_{t+1}(f(x_t, u_t))$$

- Backward propagation

$$Q_t = l(x_t, u_t) + V_{t+1}(f(x_t, u_t))$$

- Greedy optimization

$$V_t = \min_{u_t} Q_t(x_t, u_t)$$

DDP as a 2-pass algorithm

$$Q = l + V'$$

$$V = \min_u Q$$

DDP as a 2-pass algorithm

- Pass 1: back-propagate an approximation of V
  - We can solve Belman for quadratic cost and linear dynamics

- Pass 2: forward propagate gains and trajectory

DDP as a 2-pass algorithm

- Pass 1: backpropagate an approximation of V

DDP as a 2-pass algorithm

- Pass 2: forward propagate gains and trajectory

DDP as a 2-pass algorithm

- Globalization (because nonconvexity)
- Line search
  - $u = u^* + k + K\,(x\text{-}x^*)$
  - $x' = f(x,u)$

- Regularization
  - $Q_{uu} = L_{uu} + F_u^{\,T}\, V_{xx}\, F_u$
  - $k = Q_{uu}^{\ -1}\, Q_u$
  - $K = Q_{uu}^{\ -1}\, Q_{ux}$

Multiple views on DDP

3. DDP as sparse SQP

DDP as sparse SQP

$$\min_{\{x\},\{u\}} \sum_{t=0}^{T-1} l(x_t, ut) \; + l_T\,(x_T)$$

s.t. $\forall$ t = 0..T-1 $\quad x_{t+1} = f\,(x_t\,,\,u_t)$

DDP as sparse SQP

- Reminder
- Non linear problem

$$\min_{y} l(y)$$
$$\text{s.t.} \;\; f(y)=0$$

- Resulting "linearization"

$$\min_{\Delta y} l(y) + L_y \Delta y + \frac{1}{2} \Delta y^T L_{yy} \Delta y$$
$$\text{s.t.} \;\; f(y) + F_y \, \Delta y = 0$$

DDP as sparse SQP

$$\min_{\Delta y} l(y) + L_y \Delta y + \frac{1}{2} \Delta y^T L_{yy} \Delta y$$

$$\text{s.t.} \quad f(y) + F_y \Delta y = 0$$

- Lagrangian on the NLP

$$\underbrace{\mathfrak{L}}_{\text{Lagrangian}}(y, \lambda) = l(\underbrace{y}_{\text{Primal variable}}) + \underbrace{\lambda}_{\text{Dual variable (multipliers)}}{}^T f(y)$$

DDP as sparse SQP

$$\min_{\Delta y} l(y) + L_y \Delta y + \frac{1}{2} \Delta y^T L_{yy} \Delta y$$

$$\text{s.t. } f(y) + F_y \Delta y = 0$$

- Lagrangian on the QP

$$\mathfrak{L}(\Delta y, \lambda) = L_y \Delta y + \frac{1}{2} \Delta y^T L_{yy} \Delta y$$
$$+ \lambda^T (F_y \Delta y - f(y))$$

DDP as sparse SQP

$$\min_{\Delta y} l(y) + L_y \Delta y + \frac{1}{2} \Delta y^T L_{yy} \Delta y$$

$$\text{s.t. } f(y) + F_y \Delta y = 0$$

- Lagrangian on the QP

$$\mathfrak{L}(\Delta y, \lambda) = L_y \Delta y + \frac{1}{2} \Delta y^T L_{yy} \Delta y + \lambda^T (F_y \Delta y - f(y))$$

- Newton step

$$\begin{pmatrix} L_{yy} & {F_y}^T \\ F_y & 0 \end{pmatrix} \begin{pmatrix} \Delta y \\ \lambda \end{pmatrix} = \begin{pmatrix} -L_y \\ -f(y) \end{pmatrix}$$

DDP as sparse SQP

$$\min_{\{\Delta x\},\{\Delta u\}} \sum_{t=0}^{T-1} \begin{pmatrix} L_x \\ L_u \end{pmatrix}^T \begin{pmatrix} \Delta x_t \\ \Delta u_t \end{pmatrix} + \frac{1}{2} \begin{pmatrix} \Delta x_t \\ \Delta u_t \end{pmatrix}^T \begin{pmatrix} L_{xx} & L_{xu} \\ L_{ux} & L_{uu} \end{pmatrix} \begin{pmatrix} \Delta x_t \\ \Delta u_t \end{pmatrix} + \cdots$$

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

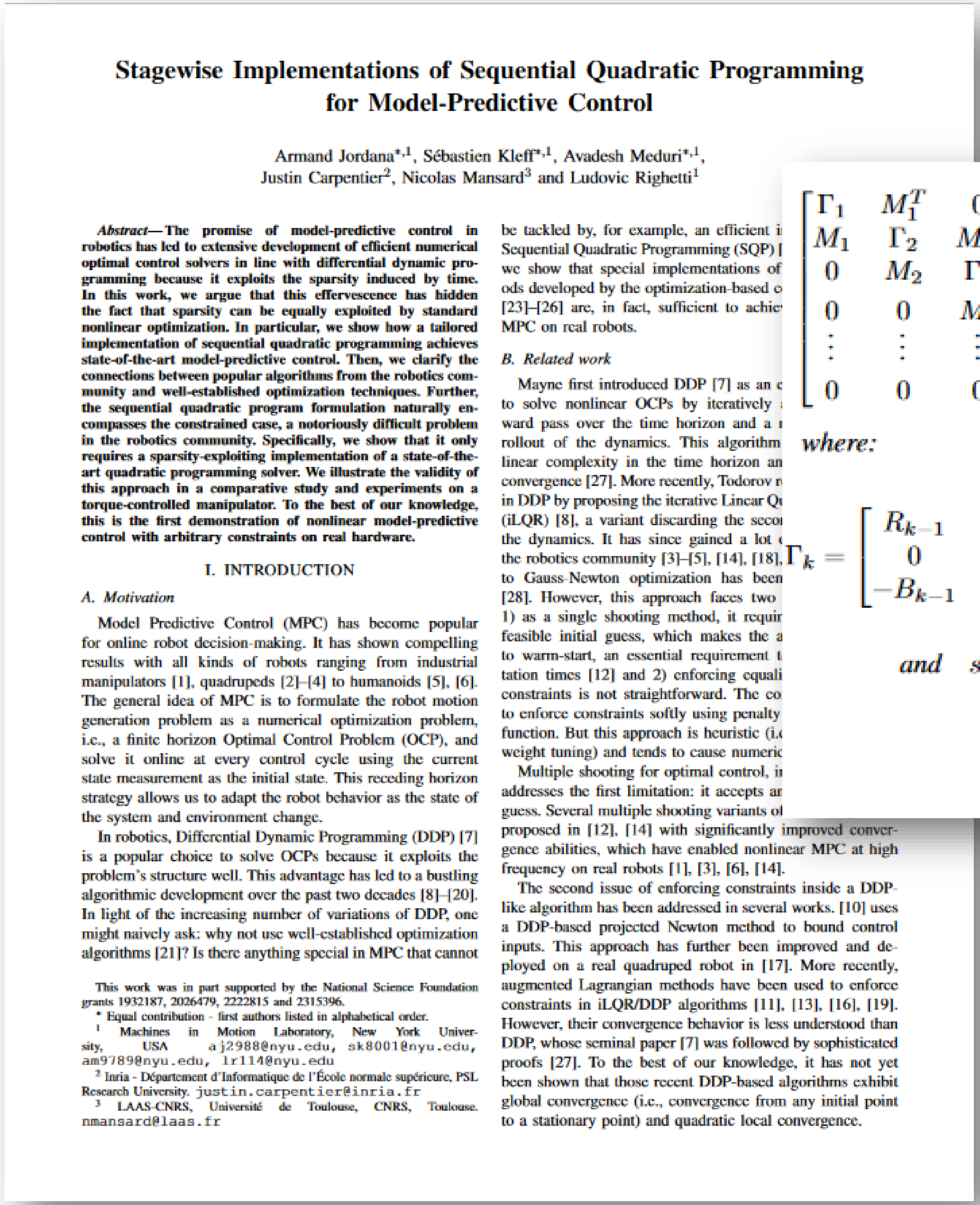
$$\begin{pmatrix} L_{yy} & {F_y}^T \\ F_y & 0 \end{pmatrix} \begin{pmatrix} \Delta y \\ \lambda \end{pmatrix} = \begin{pmatrix} -L_y \\ -f(y) \end{pmatrix}$$

DDP as sparse SQP

$$\min_{\{\Delta x\},\{\Delta u\}} \sum_{t=0}^{T-1} \begin{pmatrix} L_x \\ L_u \end{pmatrix}^T \begin{pmatrix} \Delta x_t \\ \Delta u_t \end{pmatrix} + \frac{1}{2} \begin{pmatrix} \Delta x_t \\ \Delta u_t \end{pmatrix}^T \begin{pmatrix} L_{xx} & L_{xu} \\ L_{ux} & L_{uu} \end{pmatrix} \begin{pmatrix} \Delta x_t \\ \Delta u_t \end{pmatrix} + \cdots$$

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

$$\left[\begin{array}{cccc|ccc|cccc}
L_{xx} & & & & L_{xu} & & & -I & F_x^T & & \\
& \ddots & & & & \ddots & & & \ddots & \ddots & \\
& & L_{xx} & & & & L_{xu} & & & -I & F_x^T \\
& & & L_{xx} & & & & & & & -I \\
\hline
L_{ux} & & & & L_{uu} & & & & F_u^T & & \\
& \ddots & & & & \ddots & & & & \ddots & \\
& & L_{ux} & & & & L_{uu} & & & & F_u^T \\
\hline
-I & & & & & & & & & & \\
F_x & -I & & & F_u & & & & & & \\
& \ddots & \ddots & & & \ddots & & & & & \\
& & F_x & -I & & & F_u & & & &
\end{array}\right]
\left[\begin{array}{c}
\Delta x_0 \\ \vdots \\ \Delta x_{T-1} \\ \Delta x_T \\ \hline \Delta u_0 \\ \vdots \\ \Delta u_{T-1} \\ \hline \lambda_0 \\ \lambda_1 \\ \vdots \\ \lambda_{T-1}
\end{array}\right]
= -
\left[\begin{array}{c}
L_x \\ \vdots \\ L_x \\ L_x \\ \hline L_u \\ \vdots \\ L_u \\ \hline f_0 \\ f_1 \\ \vdots \\ f_{T-1}
\end{array}\right]$$



Stagewise Implementations of Sequential Quadratic Programming for Model-Predictive Control

Armand Jordana*,1, Sébastien Kleff*,1, Avadesh Meduri*,1, Justin Carpentier[2], Nicolas Mansard[3] and Ludovic Righetti[1]

***Abstract*—The promise of model-predictive control in robotics has led to extensive development of efficient numerical optimal control solvers in line with differential dynamic programming because it exploits the sparsity induced by time. In this work, we argue that this effervescence has hidden the fact that sparsity can be equally exploited by standard nonlinear optimization. In particular, we show how a tailored implementation of sequential quadratic programming achieves state-of-the-art model-predictive control. Then, we clarify the connections between popular algorithms from the robotics community and well-established optimization techniques. Further, the sequential quadratic program formulation naturally encompasses the constrained case, a notoriously difficult problem in the robotics community. Specifically, we show that it only requires a sparsity-exploiting implementation of a state-of-the-art quadratic programming solver. We illustrate the validity of this approach in a comparative study and experiments on a torque-controlled manipulator. To the best of our knowledge, this is the first demonstration of nonlinear model-predictive control with arbitrary constraints on real hardware.**

I. INTRODUCTION

A. Motivation

Model Predictive Control (MPC) has become popular for online robot decision-making. It has shown compelling results with all kinds of robots ranging from industrial manipulators [1], quadrupeds [2]–[4] to humanoids [5], [6]. The general idea of MPC is to formulate the robot motion generation problem as a numerical optimization problem, i.e., a finite horizon Optimal Control Problem (OCP), and solve it online at every control cycle using the current state measurement as the initial state. This receding horizon strategy allows us to adapt the robot behavior as the state of the system and environment change.

In robotics, Differential Dynamic Programming (DDP) [7] is a popular choice to solve OCPs because it exploits the problem's structure well. This advantage has led to a bustling algorithmic development over the past two decades [8]–[20]. In light of the increasing number of variations of DDP, one might naively ask: why not use well-established optimization algorithms [21]? Is there anything special in MPC that cannot be tackled by, for example, an efficient i
Sequential Quadratic Programming (SQP) [
we show that special implementations of
ods developed by the optimization-based c
[23]–[26] are, in fact, sufficient to achie
MPC on real robots.

B. Related work

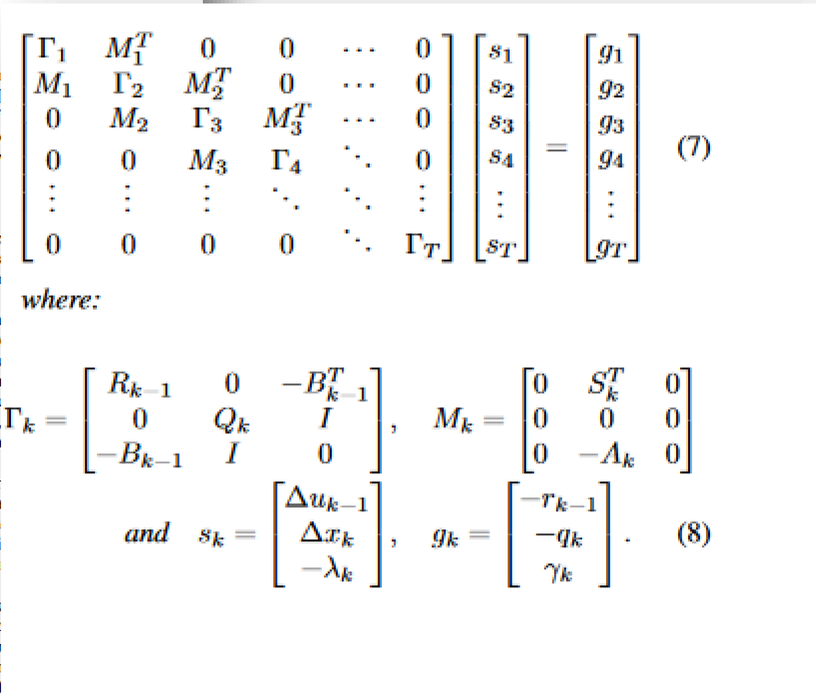
Mayne first introduced DDP [7] as an
to solve nonlinear OCPs by iteratively
ward pass over the time horizon and a
rollout of the dynamics. This algorithm
linear complexity in the time horizon an
convergence [27]. More recently, Todorov
in DDP by proposing the iterative Linear Q
(iLQR) [8], a variant discarding the seco
the dynamics. It has since gained a lot
the robotics community [3]–[5], [14], [18],
to Gauss-Newton optimization has been
[28]. However, this approach faces two
1) as a single shooting method, it requi
feasible initial guess, which makes the
to warm-start, an essential requirement
tation times [12] and 2) enforcing equali
constraints is not straightforward. The co
to enforce constraints softly using penalty
function. But this approach is heuristic (i.
weight tuning) and tends to cause numeric

Multiple shooting for optimal control, i
addresses the first limitation: it accepts a
guess. Several multiple shooting variants o
proposed in [12], [14] with significantly improved convergence abilities, which have enabled nonlinear MPC at high frequency on real robots [1], [3], [6], [14].

The second issue of enforcing constraints inside a DDP-like algorithm has been addressed in several works. [10] uses a DDP-based projected Newton method to bound control inputs. This approach has further been improved and deployed on a real quadruped robot in [17]. More recently, augmented Lagrangian methods have been used to enforce constraints in iLQR/DDP algorithms [11], [13], [16], [19]. However, their convergence behavior is less understood than DDP, whose seminal paper [7] was followed by sophisticated proofs [27]. To the best of our knowledge, it has not yet been shown that those recent DDP-based algorithms exhibit global convergence (i.e., convergence from any initial point to a stationary point) and quadratic local convergence.

This work was in part supported by the National Science Foundation grants 1932187, 2026479, 2222815 and 2315396.

* Equal contribution - first authors listed in alphabetical order.
[1] Machines in Motion Laboratory, New York University, USA aj2988@nyu.edu, sk8001@nyu.edu, am9789@nyu.edu, lr114@nyu.edu
[2] Inria - Département d'Informatique de l'École normale supérieure, PSL Research University. justin.carpentier@inria.fr
[3] LAAS-CNRS, Université de Toulouse, CNRS, Toulouse. nmansard@laas.fr



$$\begin{bmatrix} \Gamma_1 & M_1^T & 0 & 0 & \cdots & 0 \\ M_1 & \Gamma_2 & M_2^T & 0 & \cdots & 0 \\ 0 & M_2 & \Gamma_3 & M_3^T & \cdots & 0 \\ 0 & 0 & M_3 & \Gamma_4 & \ddots & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \ddots & \Gamma_T \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ \vdots \\ s_T \end{bmatrix} = \begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \\ \vdots \\ g_T \end{bmatrix} \quad (7)$$

where:

$$\Gamma_k = \begin{bmatrix} R_{k-1} & 0 & -B_{k-1}^T \\ 0 & Q_k & I \\ -B_{k-1} & I & 0 \end{bmatrix}, \quad M_k = \begin{bmatrix} 0 & S_k^T & 0 \\ 0 & 0 & 0 \\ 0 & -A_k & 0 \end{bmatrix}$$

$$\text{and} \quad s_k = \begin{bmatrix} \Delta u_{k-1} \\ \Delta x_k \\ -\lambda_k \end{bmatrix}, \quad g_k = \begin{bmatrix} -r_{k-1} \\ -q_k \\ \gamma_k \end{bmatrix}. \quad (8)$$

Sebastien Kleff

Armand Jordana

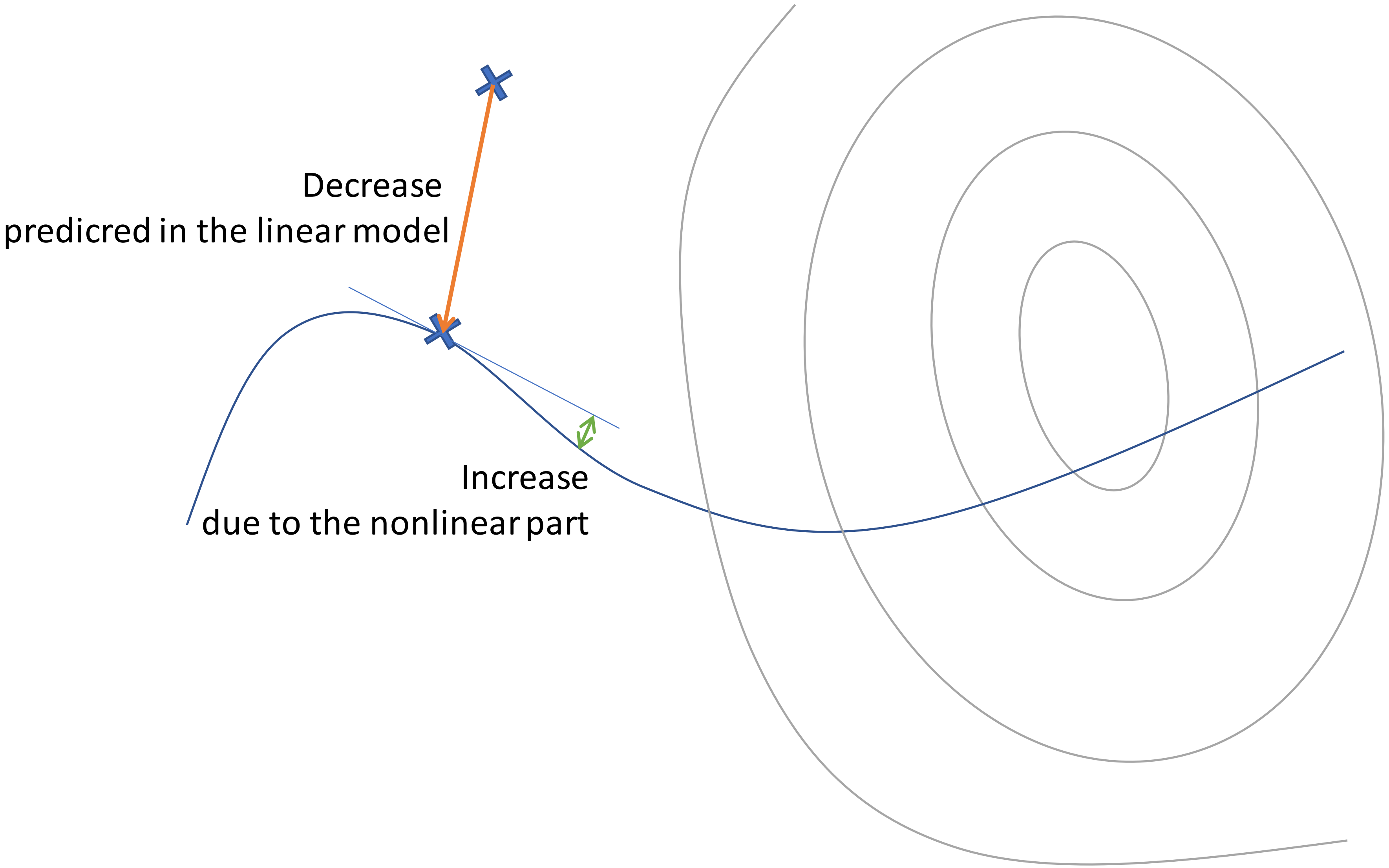
Avadesh Meduri

Rohan Budhiraja

crocoddyl

Contact Robot Optimal Control by Differential Dynamic Library

Linear versus Nonlinear Rollouts

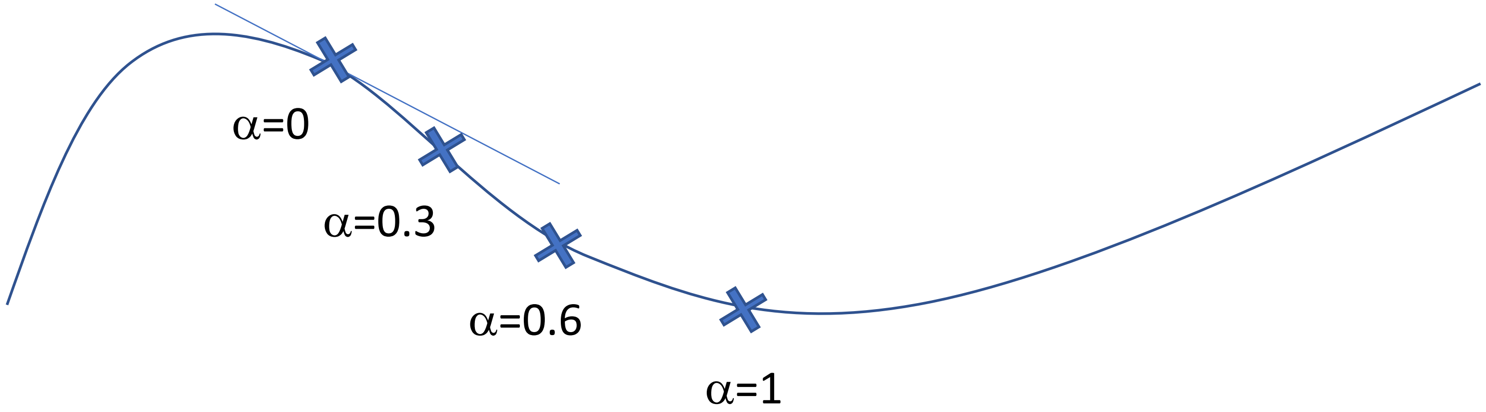


Linear versus Nonlinear Rollouts

- Case where the current guess is feasible: $x_+ = f(x,u)$

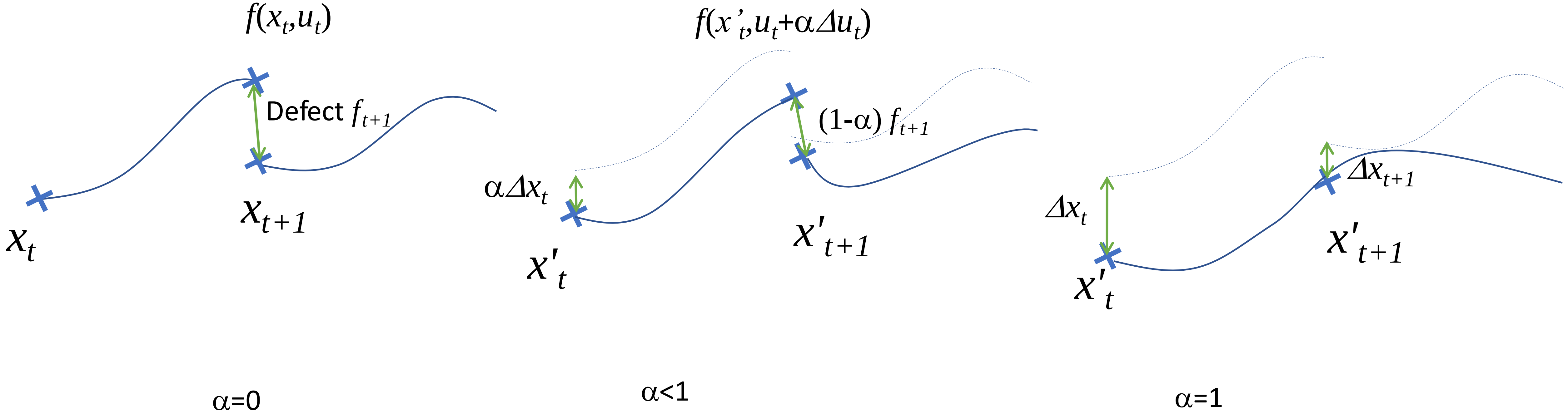
- If we know $\{u\}$, then we can get a feasible X by roll-out

$$x_+ + \Delta x := f(x, u+\alpha\,\Delta u)$$



Linear versus Nonlinear Rollouts

- Case where the current guess is not feasible: $x_{t+1} = f(x_t,u_t)+f_{t+1}$



Feasibility DDP: a (somehow) multiple-shooting algorithm

- Given the OCP formulated as an NLP

- Iterates:
  - Linearize the NLP … obtain a cQP
  - Compute a partial step using a LQR
    - Backward pass to backpropagate the Riccati gains
    - Increase regularization is Hamiltonian Hessian becomes nonpositive
  - Perfom a line-search in the direction $\Delta X, \Delta U$
    - For any step α, perform a forward propagation with gaps (defect) reduced of α
    - Accept under Wolfe conditions
  - Stop when Lagrangian gradients vanish

Policy derivatives

- At convergence,

  the derivatives of the NLP and the LCQP are the same

- The linear optimal policy (LQ regulator) is

$$\Delta\pi(\Delta x) = k + K\Delta x$$

- The derivatives of the nonlinear optimal policy is

$$\frac{\partial \pi}{\partial x} = K$$

Policy Taylor approximation

- A approximation of the optimal policy is

$$\pi(x + \Delta x) = \pi(x) + K\Delta x$$

- The derivatives of the nonlinear optimal policy is

$$\frac{\partial \pi}{\partial x} = K$$