Nonlinear optimal control of quadruped

Niraj Rathod

May 3, 2023

Chapter 1

Modelling

The centroidal dynamics of a quadruped can be expressed by,

$$m\ddot{x}_c = mg + \sum_{i=1}^{e} f_i \tag{1.1}$$

$$I_{com}\,\dot{\omega} + \omega_\times I_{com}\omega = \sum_{i=1}^{e} (x_{f_i} - x_c)_\times f_i \tag{1.2}$$

Equation 1.1 and 1.2 describe the linear and nonlinear dynamics of a robot.

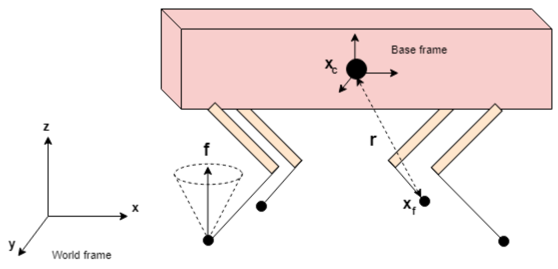

Figure 1.1: Schematic of a quadruped robot

where,

- $x_c \in \mathbb{R}^3$ is the position of a Center of Mass (**COM**)
- $\omega \in \mathbb{R}^3$ is the angular speed of a COM [1]
- $g = \begin{bmatrix} 0 & 0 & -9.8 \end{bmatrix}^T$ is a gravity vector
- $f_i \in \mathbb{R}^3$ is the ground reaction force at i^{th} leg
- $I_{com} \in \mathbb{R}^{3\times3}$ is a moment of inertia of a robot
- $x_{f_i} \in \mathbb{R}^3$ is a foot location of i^{th} leg
- e is a number of end effectors (foot) in contact

[1] $\omega_\times$ and $(x_{f_i} - x_c)_\times$ are skew-symmetric matrices for cross product

The body orientation in the body frame is described by X-Y-Z Euler angles $\varPhi = [\phi,\, \theta,\, \varPsi]^T$, with ϕ , θ and $\varPsi$ are roll, pitch and yaw angles respectively. $\dot{\varPhi}$ is related to ω (in world frame) by transformation matrix,

$$\dot{\varPhi} = E^{-1}(\varPsi, \theta)\, \omega \tag{1.3}$$

where, $E^{-1}\,(\varPsi, \theta) = \begin{bmatrix} cos(\varPsi)/cos(\theta) & sin(\varPsi)/cos(\theta) & 0 \\ -sin(\varPsi) & cos(\varPsi) & 0 \\ cos(\varPsi)\, tan(\theta) & sin(\varPsi)\, tan(\theta) & 1 \end{bmatrix}$

Defining the state vector $x = [x_c,\, \dot{x}_c,\, \varPhi,\, \omega, g]$ we have,

$$\begin{bmatrix} \dot{x}_c \\ \ddot{x}_c \\ \dot{\varPhi} \\ \dot{\omega} \\ \dot{g} \end{bmatrix} = \begin{bmatrix} 0_3 & 1_3 & 0_3 & 0_3 & 0_3 \\ 0_3 & 0_3 & 0_3 & 0_3 & 1_3 \\ 0_3 & 0_3 & 0_3 & E^{-1}(\varPsi, \theta) & 0_3 \\ 0_3 & 0_3 & 0_3 & -I_{com}^{-1}(\omega_\times I_{com}) & 0_3 \\ 0_3 & 0_3 & 0_3 & 0_3 & 0_3 \end{bmatrix} \begin{bmatrix} x_c \\ \dot{x}_c \\ \varPhi \\ \omega \\ g \end{bmatrix} + \begin{bmatrix} 0_3 & \cdots & 0_3 \\ 1_3/m & \cdots & 1_3/m \\ 0_3 & \cdots & 0_3 \\ I_{com}^{-1}(x_{f1} - x_c)_\times & \cdots & I_{com}^{-1}(x_{f_n} - x_c)_\times \\ 0_3 & \cdots & 0_3 \end{bmatrix} \begin{bmatrix} f_i \\ \vdots \\ f_n \end{bmatrix} \tag{1.4}$$

The cross product between $(x_{f_i} - x_c)$ and f_i can be written as,

$$(x_{f_i} - x_c)_\times f_i = \begin{bmatrix} 0 & -(x_{f_i}^z - x_c^z) & (x_{f_i}^y - x_c^y) \\ (x_{f_i}^z - x_c^z) & 0 & -(x_{f_i}^x - x_c^x) \\ -(x_{f_i}^y - x_c^y) & (x_{f_i}^x - x_c^x) & 0 \end{bmatrix} \begin{bmatrix} f_i^x \\ f_i^y \\ f_i^z \end{bmatrix} \tag{1.5}$$

Notice that g is added as an additional state without dynamics so to simplify the formulation. More concisely this equation can be written as,

$$\dot{x}(t) = A(\varPsi, \theta, \omega)\, x(t) + B(x_{f_i}, x_c)\, u(t) \tag{1.6}$$

1.1 Discrete time model

We can discretize the model using explicit Euler method. The accuracy of the resulting model depends on the choice of sampling time. For larger sampling more accurate method may be used.

$$x(k + T_s) = x(k) + T_s \cdot\, F(x(t), u(t)) \tag{1.7}$$

where function $F(x(t), u(t))$ is a state update equation. The discrete time model derived by explicit Euler is,

$$\begin{bmatrix} x_c(k+1) \\ v_c(k+1) \\ \varPhi(k+1) \\ \omega(k+1) \\ g(k+1) \end{bmatrix} = \begin{bmatrix} 1_3 & 1_3\, T_s & 0_3 & 0_3 & 0_3 \\ 0_3 & 1_3 & 0_3 & 0_3 & 1_3\, T_s \\ 0_3 & 0_3 & 1_3 & T_s\, E^{-1}(\varPsi(k), \theta(k)) & 0_3 \\ 0_3 & 0_3 & 0_3 & 1_3 - T_s\, I_{com}^{-1}(\omega_\times I_{com}) & 0_3 \\ 0_3 & 0_3 & 0_3 & 0_3 & 1_3 \end{bmatrix} \begin{bmatrix} x_c(k) \\ v_c(k) \\ \varPhi(k) \\ \omega(k) \\ g(k) \end{bmatrix} + \begin{bmatrix} 0_3 & \cdots & 0_3 \\ 1_3\, T_s/m & \cdots & 1_3\, T_s/m \\ 0_3 & \cdots & 0_3 \\ T_s\, I_{com}^{-1}\, (x_{f_1}(k) - x_c(k)) & \cdots & T_s\, I_{com}^{-1}\, (x_{f_n}(k) - x_c(k)) \\ 0_3 & \cdots & 0_3 \end{bmatrix} \begin{bmatrix} f_i \\ \vdots \\ f_n \end{bmatrix} \tag{1.8}$$

For ease of notation we set $v_c = \dot{x}_c$. Compactly,

$$x(k+1) = A(\Psi(k), \theta(k), \omega(k))\, x(k) + B(x_{f_i}(k), x_c(k))\, u(k) \tag{1.9}$$

The foothold locations x_{f_i} are considered as parameters for the system given by higher level trajectory optimizer.

1.2 Linearization

If we consider this nonlinear dynamical model $\dot{x}(t) = f(x(t), u(t))$. The first order approximation by Taylor series around certain points $\bar{x}$ and $\bar{u}$ is,

$$\dot{x} = f(\bar{x}, \bar{u}) + \frac{\partial f(x(t), u(t))}{\partial x(t)}\bigg|_{x=\bar{x}, u=\bar{u}} (x(t) - \bar{x}) + \frac{\partial f(x(t), u(t))}{\partial u(t)}\bigg|_{x=\bar{x}, u=\bar{u}} (u(t) - \bar{u}) \tag{1.10}$$

Further,

$$\dot{x}(t) = f(\bar{x}, \bar{u}) + A_c(x(t) - \bar{x}) + B_c(u(t) - \bar{u}) \tag{1.11}$$

Rearranging the terms,

$$\dot{x}(t) = A_c x(t) + B_c u(t) + f(\bar{x}, \bar{u}) - A_c\bar{x} - B_c\bar{u} \tag{1.12}$$

Setting $r = f(\bar{x}, \bar{u}) - A_c\bar{x} - B_c\bar{u}$ and after discretization using explicit Euler with sampling time T_s,

$$\begin{aligned} x(k+1) &= x(k) + T_s\,[A_c x(k) + B_c u(k) + r] \\ &= (I + T_s\, A_c) x(k) + T_s B_c u(k) + T_s\, r \end{aligned} \tag{1.13}$$

Mistake in previous formulation

$$x(k+1) = (T_s\, A_c) x(k) + T_s B_c u(k) + T_s\, (r + x(k)) \tag{1.14}$$

Generalizing to this equation over the linearization trajectory leads to,

$$x(k+1) = A(k) x(k) + B(k) u(k) + T_s\, r(k) \tag{1.15}$$

where, $A = T_s A_c$, $B = T_s B_c$

1.3 Stabilizing controller

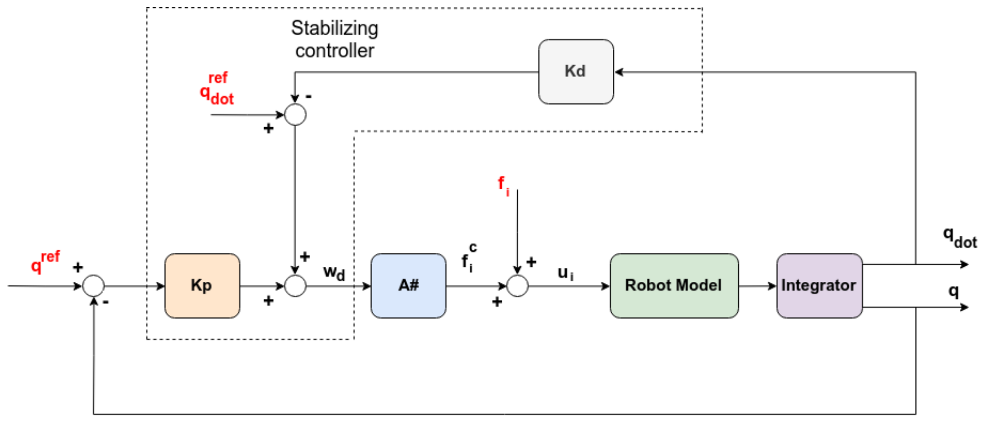

Figure 1.2: Stabilizing controller scheme

The robot model is nonlinear unstable. In ROS there is a whole body controller (PD) that takes care of the stability of the robot while standing or dynamic walk. This PD runs at higher frequency (1kHz). Additionally, there is a gravity compensation and QP controller that maps input forces calculated by whole body PD controller to the Torques setpoints for the joint PD controllers. CHECK THIS!! The stabilizing PD control is,

$$W^d = K_p(q - q_{ref}) + K_d(\dot{q} - \dot{q}_{ref}) \tag{1.16}$$

where, $q = [x_c \;\; \Phi] \in R^6$ and $K_p, K_d \in R^{6\times 6}$ are the diagonal matrices. The compensation of the ground reaction forces can be mapped by following transformation,

$$f_i^c = A^{\#} W_d \tag{1.17}$$

for $A = \begin{bmatrix} 1_3 & \cdots & 1_3 \\ (x_{f_1} - x_c)_x & \cdots & (x_{f_n} - x_c)_x \end{bmatrix}$, $A^{\#}$ is a sudo inverse of $A \in R^{12\times 6}$ and $(x_{f_i} - x_c)_x \in R^{3\times 3}$ is a skew symmetric matix. So the input to the robot model is,

$$u_i(t) = f_i(t) + f_i^c(t) \tag{1.18}$$

1.4 Validation: closed loop

For validation the data have been collected from ROS simulator with a whole body PD controller. As the system in nonlinear unstable, it is necessary to design a stabilizing PD with the centroidal dynamic model under consideration to carry out the closed loop validation. A stabilizing PD controller has been designed using the similar gains from the ROS whole body PD controller to maintain the consistency during the validation.

- The gains are: $K_p = diag(1500{*}ones(6,1))$ and $K_d = diag([200, 200, 200, 100, 100, 100])$
- Integration methods
  - Continuous time: **ode15s**
  - Discrete time: **explicit Euler with $T_s = 4$ ms**

A continuous time nonlinear model (1.4), discrete-time nonlinear model (1.8) and LTV model (1.15), have been validated and the results can be seen in the figures 1.4-1.4. A continuous time nonlinear model fit the validation data very well. As the sampling time is very low i.e. 4 ms, the discrete time nonlinear model and LTV model show good fitting with validation data. Of course the model mismatch is compensated by the PD controller and hence contributes in the better fitting. Some overshoots in the plots can be observed due the high values of proportional gains. These gains can be changed to achieve better results but as of now it seems a reasonable fitting for the scope of this work.

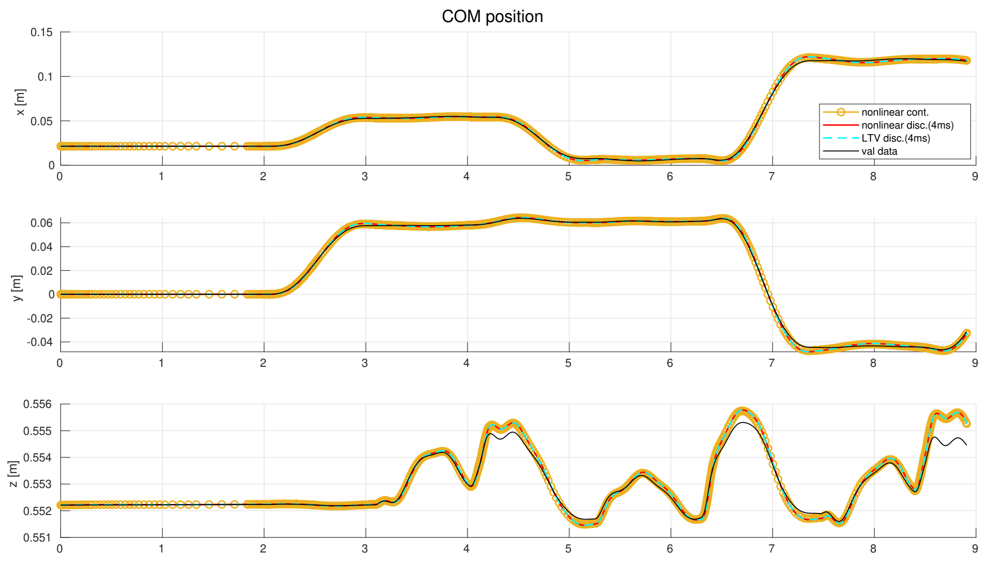

Figure 1.3: Center of mass (COM) position

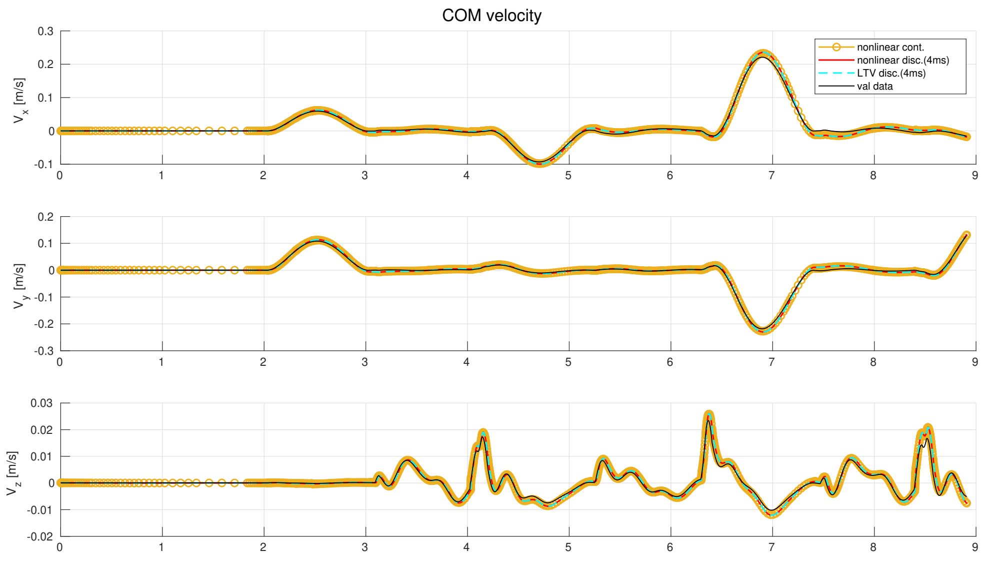

Figure 1.4: COM velocity

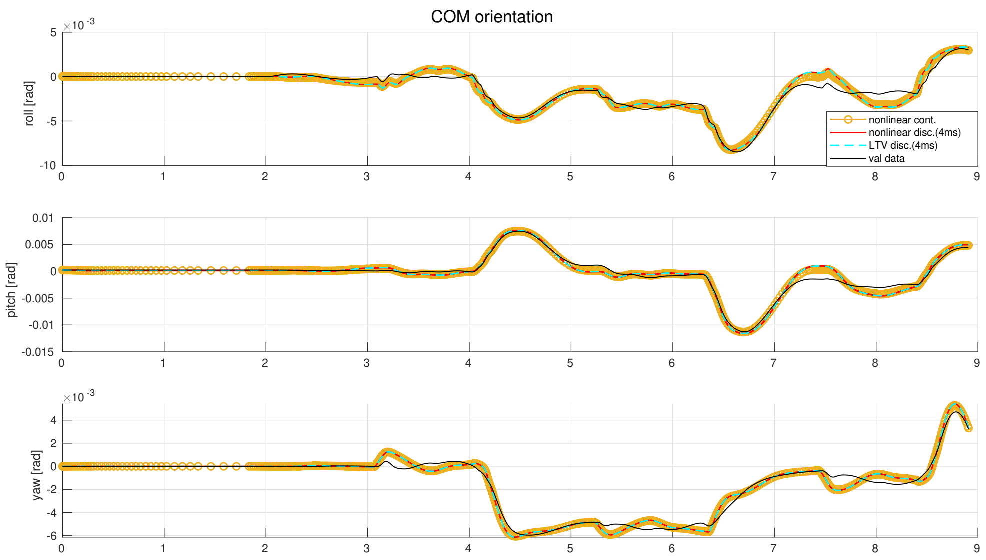

Figure 1.5: COM orientation

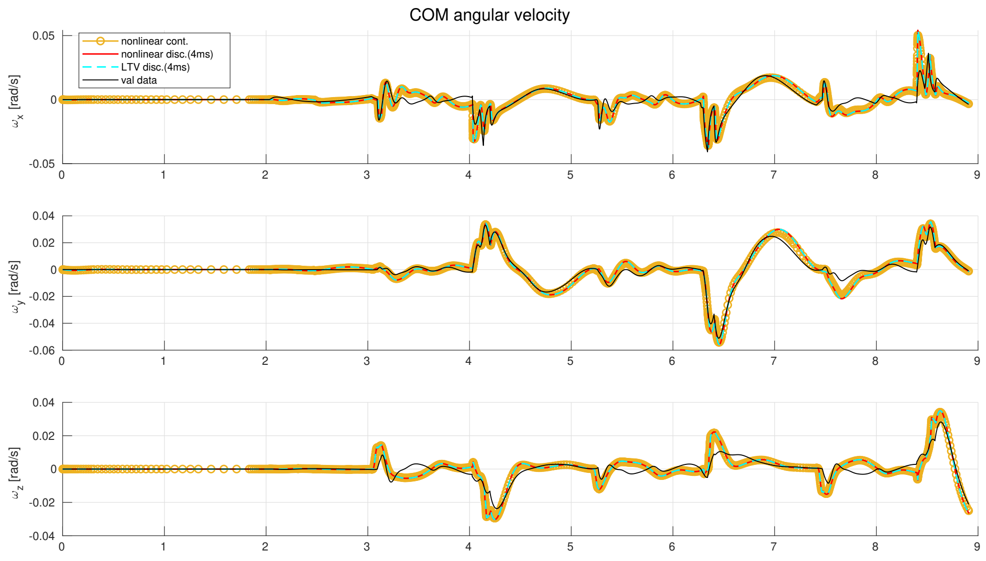

Figure 1.6: COM orientation

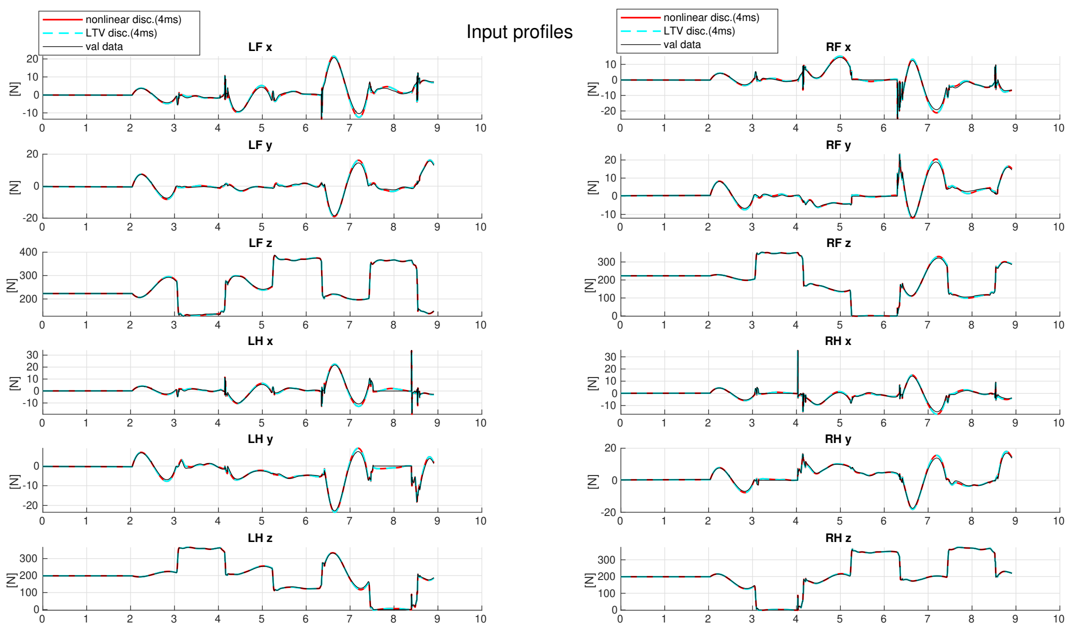

Figure 1.7: Input profile.

Similar validation with 40 ms sampling time has been carried out. The results can be seen in the following figures 1.4-1.4.

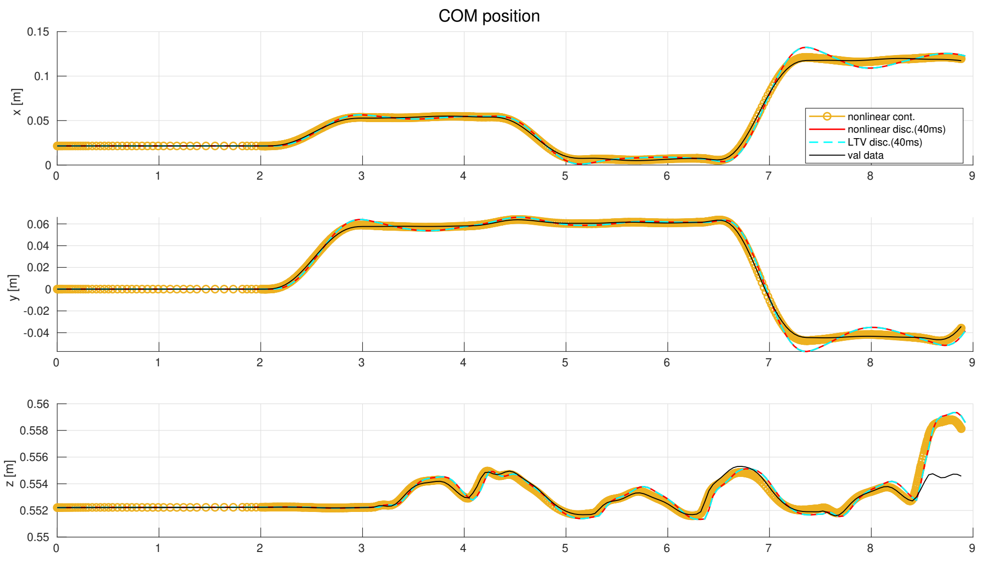

Figure 1.8: Center of mass (COM) position

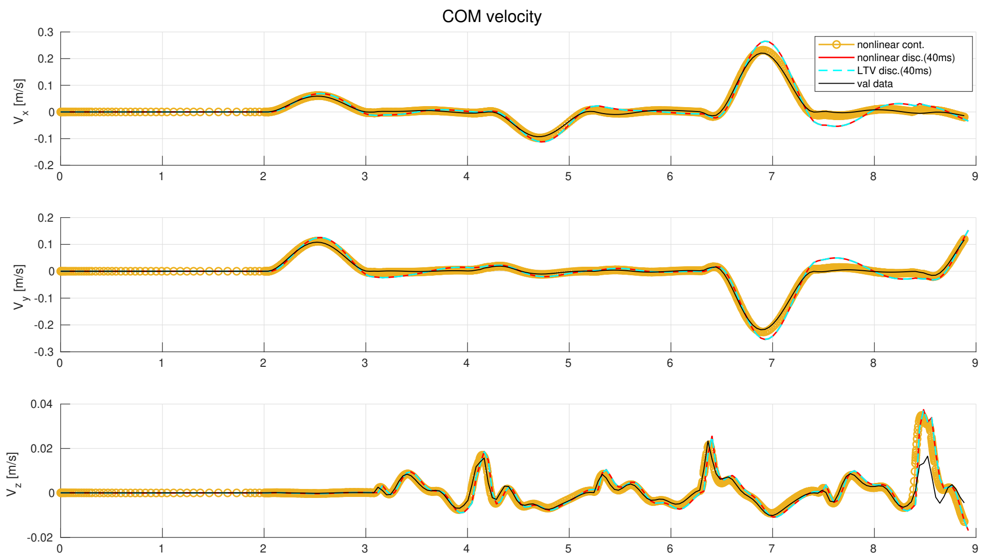

Figure 1.9: COM velocity

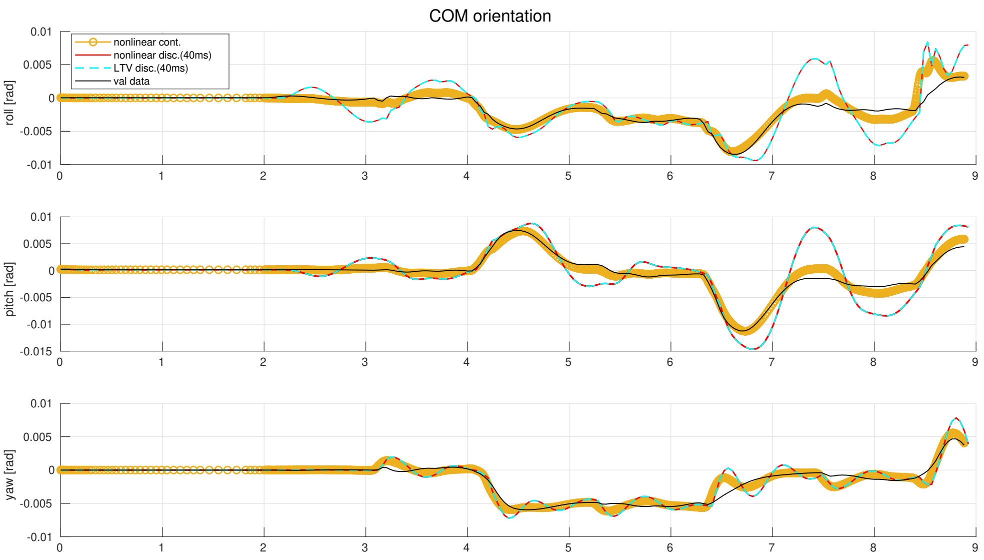

Figure 1.10: COM orientation

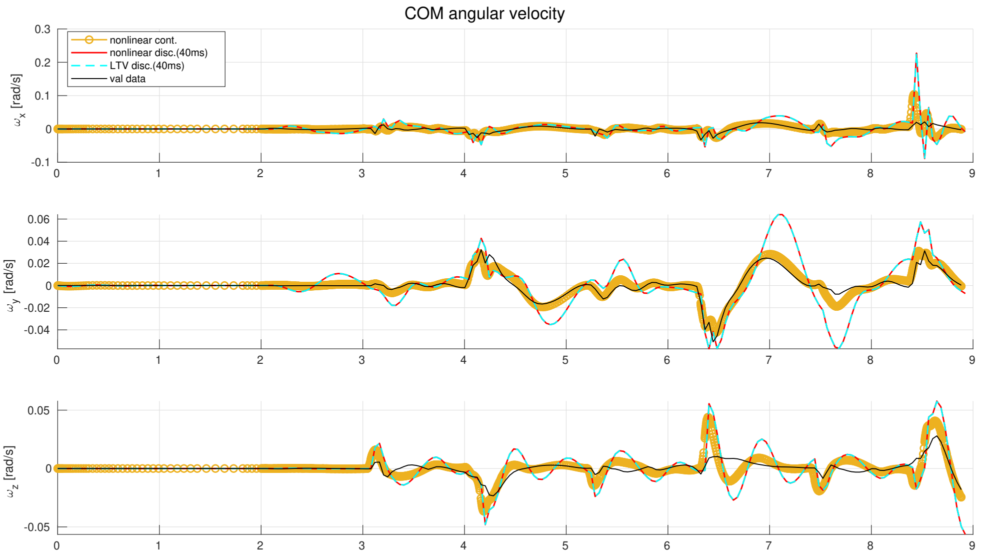

Figure 1.11: COM orientation

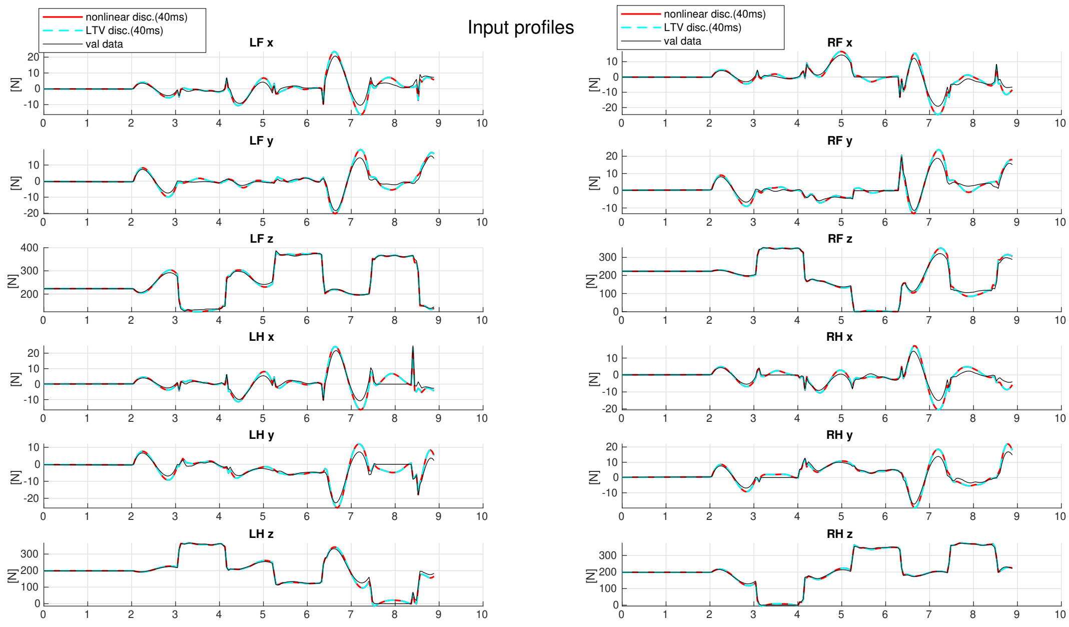

Figure 1.12: Input profile.

Chapter 2

Optimal control problem

So the optimization variables would be $\begin{bmatrix} u_0 & x_1 & \dots & u_{N-1} & x_N \end{bmatrix}$ and the optimization problem can be setup with following objective function and constraints,

$$
\begin{aligned}
\min_{x(k),u(k)} \quad & \sum_{k=0}^{N-1} Q\,(x(k) - x_{ref}(k))^2 + R\,u(k)^2 && \text{(Cost)} \\
\text{s. t.} \quad & x(k+1) = A(k)\Delta x(k) + B(k)\Delta u(k) + r(k) && \text{(Dynamics)} \\
& u_{min} \leq u_z(k) \leq u_{max} && \text{(Unilaterality)} \\
& -\mu u_z(k) \leq u_x(k) \leq \mu u_z(k) && \text{(Friction cone)} \\
& -\mu u_z(k) \leq u_y(k) \leq \mu u_z(k) && \text{(Friction cone)} \\
& D(k)\,u(k) = 0 && \text{(Complementarity)} \\
& x(0) = x_0 && \text{(Initial state)}
\end{aligned}
$$

Notice that here $x_{ref}(k)$ and $u_{ref}(k)$ are the parameters in optimization formulation. $\mu = 0.8$. The optimization was run with input tracking for prediction horizon $N = 200$ in open loop.

Chapter 3

Appendices

3.1 QP formulation

The standard QP [1] from can be written as

$$\min_{x} \quad \frac{1}{2}x^T P x + q^T x \tag{3.1a}$$

$$\text{s.t.} \quad l \leq Ax \leq u \tag{3.1b}$$

where $x \in \mathbf{R}^n$ is the optimization variable. The objective function is defined by a positive semidefinite matrix $P \in \mathbf{S}^n_+$

$$\begin{aligned} P &= diag\,(P_x, P_u) \\ Px &= diag\,(Q, Q, \cdots, Q_N) \\ Pu &= diag\,(R, \cdots, R) \end{aligned} \tag{3.2}$$

and vector $q \in \mathbf{R}^n$

$$q = \begin{bmatrix} -Q\,x_r \\ -Q\,x_r \\ \vdots \\ -Q_N\,x_r \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{3.3}$$

The linear constraints are defined by matrix $A \in \mathbf{R}^{m\times n}$

$$A_c = \left[\begin{array}{ccccc|cccc}
-I & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\
A & -I & 0 & \cdots & 0 & B & 0 & \cdots & 0 \\
0 & A & -I & \cdots & 0 & 0 & B & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \cdots & -I & 0 & 0 & \cdots & B \\
\hline
I & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\
0 & I & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\
0 & 0 & I & \cdots & 0 & 0 & 0 & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \cdots & I & 0 & 0 & \cdots & 0 \\
0 & 0 & 0 & \cdots & 0 & I & 0 & \cdots & 0 \\
0 & 0 & 0 & \cdots & 0 & 0 & I & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & I
\end{array}\right] \tag{3.4}$$

and vectors $l \in \mathbf{R}^m \cup \{-\infty\}^m$, $u \in \mathbf{R}^m \cup \{+\infty\}^m$.

$$l = \begin{bmatrix} -x_0 \\ 0 \\ \vdots \\ 0 \\ x_{min} \\ \vdots \\ x_{min} \\ u_{min} \\ \vdots \\ u_{min} \end{bmatrix} \qquad u = \begin{bmatrix} -x_0 \\ 0 \\ \vdots \\ 0 \\ x_{max} \\ \vdots \\ x_{max} \\ u_{max} \\ \vdots \\ u_{max} \end{bmatrix} \tag{3.5}$$

For the $\delta u = u_i - u_{i-1}$ regulation an additional term in the cost can be put which makes the objective function

$$\min_{x(k),u(k)} \quad \sum_{k=0}^{N-1} (x(k) - x_{ref}(k))^T Q\,(x(k) - x_{ref}(k)) + u(k)^T R\,u(k) + \tag{3.6a}$$

$$(u(k) - u(k+1))^T \Lambda (u(k) - u(k+1)) \tag{3.6b}$$

If you take only the part related to u in the cost function, it can be rearranged to,

$$\begin{aligned}
&u(k)^T R\,u(k) + (u(k) - u(k+1))^T \lambda (u(k) - u(k+1)) = \\
&u(0)^T R\,u(0) + u(1)^T R\,u(1) + u(2)^T R\,u(2) + \cdots + u(N-1)^T R\,u(N-1) \\
&+ u(0)^T \lambda u(0) + 2 * u(1)^T \lambda u(1) + 2 * u(2)^T \lambda u(2) + \cdots + 2 * u(N-2)^T \lambda u(N-2) + \\
&u(N-1)^T \lambda u(N-1) - u(1)^T \lambda u(0) - u(0)^T \lambda u(1) - u(2)^T \lambda u(1) \\
&- u(1)^T \lambda u(2) - \cdots - u(N-2)^T \lambda u(N-1) - u(N-1)^T \lambda u(N-2)
\end{aligned} \tag{3.7}$$

In vector-matrix form it can be written as,

$$u(k)^T \underbrace{\begin{bmatrix} R+\lambda & -\lambda & 0 & 0 & 0 \\ -\lambda & R+2\lambda & -\lambda & 0 & 0 \\ 0 & -\lambda & R+2\lambda & -\lambda & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & -\lambda & R+2\lambda & -\lambda \\ 0 & 0 & 0 & -\lambda & R+\lambda \end{bmatrix}}_{P_u} u(k) \tag{3.8}$$

This P_u can be used in the formulation of the standard QP for δu regulation instead of the the one in equation 3.2.

Bibliography

[1] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, “OSQP: An operator splitting solver for quadratic programs,” *ArXiv e-prints*, Nov. 2017.