Model Predictive Control of a Quadraped

CL 603 Optimization

Shubham Agrawal (180040100)
Yadul Sethi (22D0121)
Shashwat Chitransh (22M0389)
Puttu Umamaheshwara Rao (180020069)

IIT Bombay

April 10, 2023

Introduction

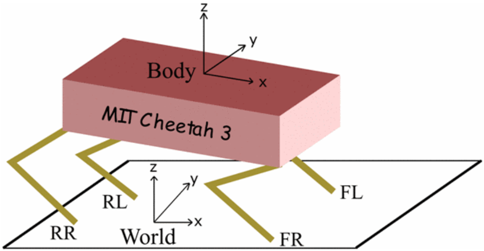

Figure 1: *Coordinate system of the Quadraped [1]*

Control Architecture of Quadraped:

- **Conventions:** Defining Body (left subscript B) and World Coordinate systems
- **State Machine:** Proposing controller based on reaction behaviors of foot placement
- **Swing Leg Control**: Accounts for the dynamics of the foot is air (which is neglected in our problem)
- **Ground Force Control:** Computing joint torques accounting for motion of body

Control System of Quadraped

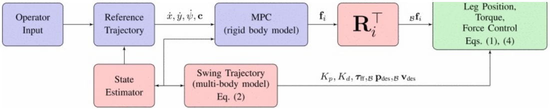

Figure 2: *Control System Block Diagram of Quadraped [1]*

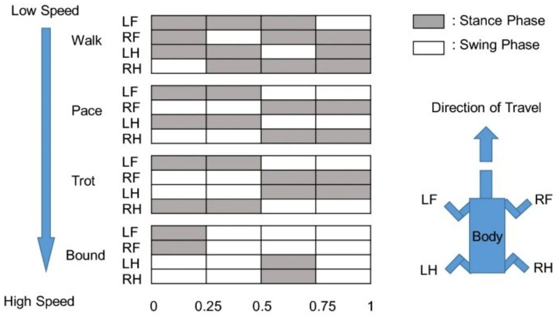

Figure 3: *Timing diagram for each leg in different gaits. The left panel illustrates orders of stance and swing phases in the typical gaits. The right one indicates the assignment of Legs layout [2]*

Simplified Robot Dynamics

- From newton's second law the acceleration of the body is given by

$$\ddot{\mathrm{p}} = \frac{\sum_{\mathrm{i=1}}^{n} \mathrm{f}_i}{m} - \mathrm{g}$$

- From newton's second law we get the rate of change of angular momentum as:

$$\frac{\mathrm{d}}{\mathrm{d}t}(\mathrm{I}\omega) = \sum_{i=1}^{n} \mathrm{r}_i \times \mathrm{f}_i \tag{1}$$

which can further expanded as

$$\frac{\mathrm{d}}{\mathrm{d}t}(\mathrm{I}\omega) = \mathrm{I}\dot{\omega} + \omega \times (\mathrm{I}\omega) \approx \mathrm{I}\dot{\omega} \tag{2}$$

where:

$$\hat{\mathrm{I}} = \mathrm{R}_z(\psi)_{\mathcal{B}}\mathsf{I}\mathrm{R}_z(\psi)^{\mathrm{T}}$$

- $R_n(\alpha)$ represents a positive rotation of α about the n − axis

Model Predictive Control

$$\dot{\mathrm{x}}(t) = \mathrm{A}_c(\psi)\mathrm{x}(t) + \mathrm{B}_c(\mathrm{r}_1,\ \ldots,\ \mathrm{r}_n,\ \psi)\mathrm{u}(t) \tag{3}$$

$$\min_{\mathrm{x},\mathrm{u}} \quad \sum_{i=0}^{k-1} \|\mathrm{x}_{i+1} - \mathrm{x}_{i+1,\mathrm{ref}}\|_{\mathrm{Q}_i} + \|\mathrm{u}_i\|_{\mathrm{R}_i} \tag{4}$$

$$\text{subject to} \quad \mathrm{x}_{i+1} = \mathrm{A}_i\mathrm{x}_i + \mathrm{B}_i\mathrm{u}_i, i = 0 \ldots k-1 \tag{5}$$

$$\underline{\mathrm{c}}_i \leq \mathrm{C}_i\mathrm{u}_i \leq \overline{\mathrm{c}}_i, i = 0 \ldots k-1 \tag{6}$$

$$\mathrm{D}_i\mathrm{u}_i = 0, i = 0 \ldots k-1 \tag{7}$$

Optimisation Problem

$$\min_{\mathrm{x}} \quad q(\mathrm{x}) = \frac{1}{2}\mathrm{x}^T G \mathrm{x} + \mathrm{x}^T d$$
$$\text{s.t} \quad [a^i]^T \mathrm{x} = b_i, \quad i \in \mathcal{E}$$
$$[a^i]^T \mathrm{x} \geq b_i, \quad i \in \mathcal{I}$$

$$A = \begin{bmatrix} [a^1]^T & \longrightarrow \\ [a^2]^T & \longrightarrow \\ \dots & \dots \\ \dots & \dots \\ [a^m]^T & \longrightarrow \end{bmatrix}_{m\times n}, \; [G]_{n\times n} \text{ symmetric}, \; [b]_{n\times 1}, \; [x]_{n\times 1}, \; [d]_{n\times 1}$$

Optimisation Problem

Converting the objective function(4) into standard form

$$G = \text{Hessian}\left(\sum_{i=0}^{k-1} ||\mathrm{x}_{i+1} - \mathrm{x}_{i+1,\mathrm{ref}}||\mathrm{Q}_i + ||\mathrm{u}_i||\mathrm{R}_i, \mathrm{x}\right)$$

$$d = \text{Grad}\left(\sum_{i=0}^{k-1} ||\mathrm{x}_{i+1} - \mathrm{x}_{i+1,\mathrm{ref}}||\mathrm{Q}_i + ||\mathrm{u}_i||\mathrm{R}_i, \mathrm{x}\right)$$

where, $\mathrm{x} = \begin{bmatrix} \mathrm{x} \\ \mathrm{u} \end{bmatrix}$

Now the objective function(4) is converted into standard form

Converting Equality Constraint to Standard Form

The equality constraints is given from the equation (6)

$$
\begin{gathered}
\mathrm{x}_{i+1} = \mathrm{A}_i \mathrm{x}_i + \mathrm{B}_i \mathrm{u}_i, \quad i = 0 \ldots k-1 \\
i = 0, \quad \mathrm{x}_1 - \mathrm{B}_0 \mathrm{u}_0 = \mathrm{A}_0 \mathrm{x}_0 \\
i = 1..k-1, \quad \mathrm{x}_{i+1} - \mathrm{A}_i \mathrm{x}_i - \mathrm{B}_i \mathrm{u}_i = 0
\end{gathered}
$$

$$
\mathrm{A}_{E_1} = \begin{bmatrix} \textit{coefficient in equation } i = 0 \\ \textit{coefficient in equation } i = 1 \\ .... \\ .... \\ \textit{coefficient in equation } i = k-1 \end{bmatrix}, \ \mathrm{b}_{E_1} = \begin{bmatrix} \mathrm{A}_0 \mathrm{x}_0 \\ 0 \\ .... \\ .... \\ 0 \end{bmatrix} \tag{11}
$$

Converting Equality Constraint to Standard Form

The third equality constraints is given from the equation (7)

$$\mathrm{D}_i \mathrm{u}_i = 0, i = 0 \dots k - 1$$

D_i contains coefficient of those u_i which are zero

$$\mathrm{A}_{E_2} = [\mathrm{D}_i], \ \mathrm{b}_{\mathrm{E}_2} = [0]$$

Converting Inequality Constraint to Standard Form

The inequality constraints is given from the equation (6)

$$\underline{\mathrm{c}}_i \leq \mathrm{C}_i \mathrm{u}_i \leq \overline{\mathrm{c}}_i, i = 0 \ldots k-1$$

u_i is defined by movement of the four legs (named FL, FR, RL ,RR) as shown in figure 1

$$\mathrm{u}_i = \begin{bmatrix} FL \\ FR \\ RL \\ RR \end{bmatrix} \Longrightarrow \begin{bmatrix} (f_\mathrm{x})_{FL} \\ (f_\mathrm{y})_{FL} \\ (f_\mathrm{z})_{FL} \\ \ldots\ldots\ldots \\ \ldots\ldots\ldots \\ (f_\mathrm{x})_{RR} \\ (f_\mathrm{y})_{RR} \\ (f_\mathrm{z})_{RR} \end{bmatrix}_{12\times1}, \qquad \begin{aligned} f_{\min} \leq f_z &\leq f_{\max} \\ -\mu f_z \leq \pm f_x &\leq \mu f_z \\ -\mu f_z \leq \pm f_y &\leq \mu f_z \end{aligned}$$

Inequality Constraints for One Leg

- The first constraint is given by

$$f_{\text{min}} \leq f_z \leq f_{\text{max}}$$

The above equation can be split into two which are given by

$$f_z \geq f_{\text{min}}$$
$$-f_z \geq -f_{\text{max}}$$

- The Second constraint is given by

$$-\mu f_z \leq \pm f_x \leq \mu f_z$$

The above equation can be split into two which are given by

$$f_x + \mu f_z \geq 0$$
$$-f_x + \mu f_z \geq 0$$

- The Third Constraint is similar to the second one except x replaced by y

Inequality Constraints for One Leg

$$\mathrm{A}_I = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & -1 \\ 1 & 0 & \mu \\ -1 & 0 & \mu \\ 0 & 1 & \mu \\ 0 & -1 & \mu \end{bmatrix}, \ \mathrm{b}_I = \begin{bmatrix} f_{\mathrm{min}} \\ -f_{\mathrm{max}} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \ \mathrm{A} = \begin{bmatrix} \mathrm{A}_{E_1} \\ \mathrm{A}_{E_2} \\ \mathrm{A}_I \end{bmatrix}, \ \mathrm{b} = \begin{bmatrix} \mathrm{b}_{E_1} \\ \mathrm{b}_{E_2} \\ \mathrm{b}_I \end{bmatrix}$$

where $\mathrm{A}_{\mathrm{E}_1}$ is given from the equation (11)

Number of Constraints and Variables

For simplicity $\mathrm{k} = 1$ the objective function in equation (4) can be re-written as

$$\min_{\mathrm{x}_1, \mathrm{u}_0} ||\mathrm{x}_1 - \mathrm{x}_{1,\mathrm{ref}}||_{\mathrm{Q}_1} + ||\mathrm{u}_0||_{\mathrm{R}_1} \Longrightarrow \min_{\mathrm{x}} \quad q\left(\mathrm{x}\right) = \frac{1}{2}\mathrm{x}^T G \mathrm{x} \ + \mathrm{x}^T d$$

where $\mathrm{x} = \begin{bmatrix} [\mathrm{x}_1]_{12\times 1} \\ [\mathrm{u}_0]_{12\times 1} \end{bmatrix}_{24\times 1}$

- Equality Constraints

$$\mathrm{x}_1 = \mathrm{A}_0\mathrm{x}_0 + \mathrm{B}_0\mathrm{u}_0, \text{ gives 12 constraints}$$

Number of Constraints and Variables

- Inequality Constraints

$$\underline{\mathrm{c}}_0 \le \mathrm{C}_0\mathrm{u}_0 \le \overline{\mathrm{c}}_0$$

$$\mathrm{u}_0 = \begin{bmatrix} FL \\ FR \\ RL \\ RR \end{bmatrix} \Longrightarrow \begin{bmatrix} (f_\mathrm{x})_{FL} \\ (f_\mathrm{y})_{FL} \\ (f_\mathrm{z})_{FL} \\ \cdots\cdots \\ \cdots\cdots \\ \cdots\cdots \\ (f_\mathrm{x})_{RR} \\ (f_\mathrm{y})_{RR} \\ (f_\mathrm{z})_{RR} \end{bmatrix}_{12\times 1} \quad \begin{array}{l} ,\ \ f_{\min} \le f_z \le f_{\max}, 2 \text{ const.} \times 4 \text{ legs} \\ -\mu f_z \le \pm f_x \le \mu f_z, 4 \text{ const.} \times 4 \text{ legs} \\ -\mu f_z \le \pm f_y \le \mu f_z, 4 \text{ const.} \times 4 \text{ legs} \end{array}$$

- The above set of equations gives a total of 40 constraints

Number of Constraints and Variables

- From the third equality constraint we get

$$\mathrm{D}_0 \mathrm{u}_0 = 0$$

$$\mathrm{u}_0 = \begin{bmatrix} FL \\ FR \\ RL \\ RR \end{bmatrix} \Longrightarrow \begin{bmatrix} (f_\mathrm{x})_{FL} \\ (f_\mathrm{y})_{FL} \\ (f_\mathrm{z})_{FL} \\ \cdots\cdots \\ \cdots\cdots \\ \cdots\cdots \\ (f_\mathrm{x})_{RR} \\ (f_\mathrm{y})_{RR} \\ (f_\mathrm{z})_{RR} \end{bmatrix}_{12\times 1}, \text{ OnGround} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}_{4\times 1}$$

- Number of constraints for this equality constraint $=$ Number of zeros in the on ground vector $\times$ 3 which varies b/w 0 & 3 (3 in this Case)

Therefore **Total Number of Constraints** lie b/w 52 & 55

Constraint Removal

- As the number of constraints vary from 52 to 55 and size of x is 24, so Working Set of A matrix may not have a full row rank
- We need to eliminate few redundant constraints
  Stage 1 Removal:

$$-\mu f_z \leq \pm f_x \leq \mu f_z, \text{4 constraints} \times \text{4 legs}$$
$$-\mu f_z \leq \pm f_y \leq \mu f_z, \text{4 constraints} \times \text{4 legs}$$
$$\Downarrow$$
$$-\mu f_z \leq f_x \leq \mu f_z, \text{2 constraints} \times \text{4 legs}$$
$$-\mu f_z \leq f_y \leq \mu f_z, \text{2 constraints} \times \text{4 legs}$$

  - In this we have removed 16 constraints
- ∴ The new constraint range is [36, 39]

Constraint Removal

Stage 2 Removal:

- From the third equality constraint we get

$$\mathrm{D}_0 \mathrm{u}_0 = 0$$

$$\mathrm{u}_0 = \begin{bmatrix} FL \\ FR \\ RL \\ RR \end{bmatrix} \Longrightarrow \begin{bmatrix} (f_\mathrm{x})_{FL} \\ (f_\mathrm{y})_{FL} \\ (f_\mathrm{z})_{FL} \\ \dots\dots \\ \dots\dots \\ \dots\dots \\ (f_\mathrm{x})_{RR} \\ (f_\mathrm{y})_{RR} \\ (f_\mathrm{z})_{RR} \end{bmatrix}_{12\times 1}, \; \text{OnGround} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}_{4\times 1}$$

- f_x,f_y,f_z will be zero for the leg which is not on the ground ($[Onground] = [0]$)

Constraint Removal

- The leg for which f_x,f_y,f_z are zero, we can remove the inequality constraints

$$
\begin{aligned}
f_{\min} \leq f_z \leq f_{\max} \\
-\mu f_z \leq f_x \leq \mu f_z \\
-\mu f_z \leq f_y \leq \mu f_z
\end{aligned}
$$

- These are 6 equations, $\therefore$ Number of removed constraints = Number of zeros in [Onground] vector $\times$ 6
- For this case Number of removed constraints = $1 \times 6 = 6$
- $\therefore$ The new constraint range is [30, 33]

Constraint Removal

Stage 3 Removal:

$$
\begin{aligned}
f_{\min} \le f_z \le f_{\max} \\
-\mu f_z \le f_x \le \mu f_z \\
-\mu f_z \le f_y \le \mu f_z
\end{aligned}
$$

$$\Downarrow$$

$$
\begin{aligned}
f_z &\ge f_{\min} \\
-f_z &\ge -f_{\max} \\
f_x + \mu f_z &\ge 0 \\
-f_x + \mu f_z &\ge 0 \\
f_y + \mu f_z &\ge 0 \\
-f_y + \mu f_z &\ge 0
\end{aligned}
$$

Constraint Removal

$$\mathrm{A}_I = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & -1 \\ 1 & 0 & \mu \\ -1 & 0 & \mu \\ 0 & 1 & \mu \\ 0 & -1 & \mu \end{bmatrix}, \ \mathrm{b} = \begin{bmatrix} \mathrm{b}_{E_1} \\ \mathrm{b}_{E_2} \\ \mathrm{b}_I \end{bmatrix}$$

- Rank of A_I matrix is three therefore there are only three independent constraints and Constraint 1 & 2 cannot become active at the same time
- It may happen that all the last four constraints can become active at the same time So we need to remove two constraints from the last four constraints
- The last four constraints are

$$-\mu f_z \leq f_x \leq \mu f_z$$
$$-\mu f_z \leq f_y \leq \mu f_z$$

Constraint Removal

- We add $+\epsilon$ and -ϵ to both the constraints such that the equation becomes

$$-\mu f_z + \epsilon \leq f_x \leq \mu f_z - \epsilon$$
$$-\mu f_z + \epsilon \leq f_y \leq \mu f_z - \epsilon$$

- With this only 2 out of 4 constraints will be present in the working set
- Total number of constraints removed $= 2 \times 4$ legs $= 8$
- $\therefore$ The new constraint range is [22,25]

Initial Guess for the MPC optimisation

- Instead of solving separate problem for calculating initial guess we have written simple code which generates a guess satisfying all the constraints
- This has been illustrated in the following example:

$$
\begin{aligned}
\mathrm{x}_1 = \mathrm{A}_0\mathrm{x}_0 + \mathrm{B}_0\mathrm{u}_0 \\
f_{\text{min}} \leq f_z \leq f_{\text{max}} \\
-\mu f_z \leq \pm f_x \leq \mu f_z \\
-\mu f_z \leq \pm f_y \leq \mu f_z
\end{aligned}
$$

- As f_{min} and f_{max} are positive we choose $f_z = \frac{f_{\text{min}}+f_{\text{max}}}{2}$, $f_x = 0, f_y = 0$
- Considering $f_{\text{min}} = 10$ & $f_{\text{max}} = 666$, then $f_z = 338$
- This modifies our u_0 matrix which is shown below

Initial Guess for MPC Optimisation

$$\text{Initial guess for } \mathrm{u}_0 = \begin{bmatrix} (f_\mathrm{x})_{FL} \\ (f_\mathrm{y})_{FL} \\ (f_\mathrm{z})_{FL} \\ \ldots\ldots\ldots \\ \ldots\ldots\ldots \\ \ldots\ldots\ldots \\ (f_\mathrm{x})_{RR} \\ (f_\mathrm{y})_{RR} \\ (f_\mathrm{z})_{RR} \end{bmatrix}_{12\times 1} \implies \begin{bmatrix} 0 \\ 0 \\ 338 \\ \ldots\ldots\ldots \\ \ldots\ldots\ldots \\ \ldots\ldots\ldots \\ 0 \\ 0 \\ 338 \end{bmatrix}_{12\times 1}$$

- x_0 is the starting position of the quadraped which is taken as $[0]_{12\times 1}$
- Initial guess for x_1 is calculated using $\mathrm{x}_1 = \mathrm{A}_0\mathrm{x}_0 + \mathrm{B}_0\mathrm{u}_0$
- In this way we have generated the initial guess which satisfies all the constraints

Active Set Approach

Below is code snippet taken from our `Active_set.m` code [3]

```
x(:,kk) = xinitial;
z = A*x(:,kk)-b;
W = find(abs(z)<=0.0000001);
tW = [1:m];
```

$$\min_{\mathrm{p}} \quad \frac{1}{2}\mathrm{p}^T G \mathrm{p} + \mathrm{p}^T g^k$$

$$\text{s.t} \quad [a^i]^T \mathrm{p} = 0, \quad i \in \mathcal{W}_k$$

QP Formulation

We remove the dynamics constraints and state trajectory from the constraints and optimization variables and include them in the cost function by the condensed approach as discussed in [4]

- The formulation changes to the following form after condensed approach

$$\mathrm{X} = \mathrm{A}_{\mathrm{qp}}\mathrm{x}_{\mathbf{0}} + \mathrm{B}_{\mathrm{qp}}\mathrm{U}$$

We describe now a brief procedure of how we have obtained the condensed formulation

- The state variables are eliminated from the decision variables from the objective function by expressing them as the function of current state and input sequence

$$\mathrm{X} = \mathrm{A}_{\mathrm{qp}}\hat{\mathrm{x}_0} + \mathrm{B}_{\mathrm{qp}}\mathrm{u} \tag{7}$$

QP Formulation

The matrix A_{qp} and B_{qp} are given by

$$\mathrm{x} \;:= [x_0^T x_1^T x_2^T \dots x_{N-1}^T x_N^T]^T, \mathrm{u} \;:= [u_0^T u_1^T u_2^T \dots u_{N-2}^T x_{N-1}^T]^T$$

$$\mathrm{A}_{\mathrm{qp}} := \begin{bmatrix} I_n \\ A \\ A^2 \\ \vdots \\ A^{N-1} \\ A^N \end{bmatrix}, \; \mathrm{B}_{\mathrm{qp}} := \begin{bmatrix} 0 & & & & \\ B & 0 & \ddots & & \\ AB & B & \ddots & & \\ \vdots & & & & \\ A^{N-2}B & & & B & 0 \\ A^{N-1}B & A^{N-2}B & \dots & AB & B \end{bmatrix}$$

QP Formulation

For our case the matrices gets simplified as

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} A_0 & & & \\ A_1 & A_0 & & \\ A_2 & A_1 & A_0 & \\ A_3 & A_2 & A_1 & A_0 \end{bmatrix}_{39\times 13} \begin{bmatrix} x_0 \end{bmatrix}_{13\times 1} \quad +$$

$$\begin{bmatrix} B_0 & 0 & 0 & 0 \\ A_1 B_0 & B_1 & 0 & 0 \\ A_2 A_1 B_0 & A_2 B_1 & B_2 & 0 \\ A_3 A_2 A_1 B_0 & A_3 A_2 B_1 & A_3 B_2 & B_2 \end{bmatrix}_{39\times 36} \begin{bmatrix} U_0 \\ U_1 \\ U_2 \\ U_3 \end{bmatrix}$$

QP Formulation

The dynamics of the system can be represented as

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\hat{\Theta}\\ \hat{\mathrm{p}}\\ \hat{\omega}\\ \dot{\hat{\mathrm{p}}}\end{bmatrix} = \begin{bmatrix}0_3 & 0_3 & \mathrm{R}_z(\psi) & 0_3\\ 0_3 & 0_3 & 0_3 & 1_3\\ 0_3 & 0_3 & 0_3 & 0_3\\ 0_3 & 0_3 & 0_3 & 0_3\end{bmatrix}\begin{bmatrix}\hat{\Theta}\\ \hat{\mathrm{p}}\\ \hat{\omega}\\ \dot{\hat{\mathrm{p}}}\end{bmatrix} + \begin{bmatrix}0_3 & \cdots & 0_3\\ 0_3 & \cdots & 0_3\\ \hat{\mathrm{I}}^{-1}[\mathrm{r}_1]_\times & \cdots & \hat{\mathrm{I}}^{-1}[\mathrm{r}_n]_\times\\ 1_3/m & \cdots & 1_3/m\end{bmatrix}\begin{bmatrix}\mathrm{f}_1\\ \vdots\\ \mathrm{f}_n\end{bmatrix} + \begin{bmatrix}0\\ 0\\ 0\\ \mathrm{g}\end{bmatrix}$$

Then we can obtain state transition equation from

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\mathrm{x}\\ \mathrm{u}\end{bmatrix} = \begin{bmatrix}\mathrm{A} & \mathrm{B}\\ 0 & 0\end{bmatrix}\begin{bmatrix}\mathrm{x}\\ \mathrm{u}\end{bmatrix}$$

Considering u as matrix of ones i.e, the legs are on ground at all times we get

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\mathrm{x}\\ 1\end{bmatrix} = \begin{bmatrix}\mathrm{A} & \mathrm{B}\\ 0 & 0\end{bmatrix}\begin{bmatrix}\mathrm{x}\\ 1\end{bmatrix}$$

QP Formulation

This simplifies as

$$\dot{x} = Ax + b \begin{bmatrix} 0 \\ 0 \\ 0 \\ g \end{bmatrix}$$

The final QP formulation turns out as

$$\begin{aligned} \min_{u} \quad & \frac{1}{2} U^T H U + U^T g \\ \text{s. t.} \quad & \underline{c} \le CU \le \bar{c} \end{aligned}$$

where c is the constant matrix

$$\begin{aligned} H &= 2(B_{qp}^T L B_{qp} + K) \\ g &= 2B_{qp}^T L(A_{qp} x_0 - y) \end{aligned}$$

Simulink Model

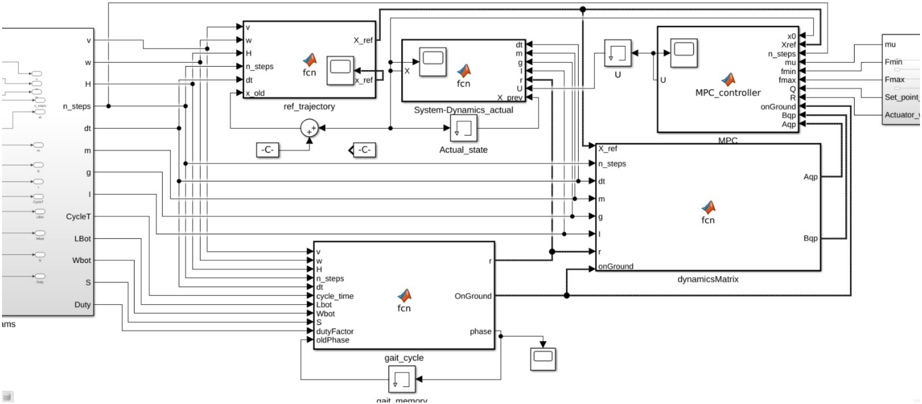

Figure 4: *Simulink model for the MPC [3]*

References I

[1] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic Locomotion in the MIT Cheetah 3 Through Convex Model-Predictive Control," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, ISSN: 2153-0866, Oct. 2018, pp. 1–9. DOI: 10.1109/IROS.2018.8594448.

[2] *Generalization of movements in quadruped robot locomotion by learning specialized motion data — ROBOMECH Journal — Full Text*, [Online]. Available: https://robomechjournal.springeropen.com/articles/10.1186/s40648-020-00174-1.

[3] *Shubhamagr281999/Quadruped_mpc_control_optimisation_cl603: This repo is collaborative work of Mr. Shubham, Mr. Yadul, Mr. Shashwat and Mr. Uma Mahesh under the course CL603 IITB fall 2023.* en. [Online]. Available: https://github.com/shubhamagr281999/Quadruped_MPC_Control_Optimisation_CL603.

References II

[4] *A condensed and sparse QP formulation for predictive control — IEEE Conference Publication — IEEE Xplore*, [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6160293.

Thank you !

Appendix - Active Set Code I

```
function U = Active_set(x0,Xref,n_steps,mu,fmin,fmax,Q,R,onGround,Bi
    %initial guess
    Xref_=reshape(Xref,1,[])';
    xinitial=zeros(24*n_steps,1);
    xPrev=x0;
    for i=0:(n_steps-1)
        for j=0:3
            if(onGround(j+1,i+1)==1)
                xinitial((12*n_steps+i*12+j*3+1):(12*n_steps+i*12+j*3
                =[0;0;(fmin+fmax)*0.5];
            end
        end
        xinitial((i*12+1):(i*12+12))=Ai(:,:,i+1)*xPrev+Bi(:,:,i+1)*..
        [xinitial((12*(i+n_steps)+1):(12*(i+n_steps)+12));1];
        xPrev=xinitial((i*12+1):(i*12+12));
    end

    onGround=reshape(onGround,1,[])';
    to_keep_constraint=zeros(6*size(onGround,1),1);
    cons_count=0;
    for i=1:int8(size(onGround,1))
        if(onGround(i)==1)
```

Appendix - Active Set Code II

```
            to_keep_constraint((cons_count+1):(cons_count+6),1)...
            =((i-1)*6+1):6*i;
            cons_count=cons_count+6;
        end
    end

    to_keep_constraints=to_keep_constraint(1:cons_count,1);
    [Du,C,Du_E,C_E] = equality_constraint();
    [I,b_I] = Inequality_cons();
    m_eq=size(Du,1)+size(C,1);
    A = [Du;C;I];
    b = [Du_E;C_E;b_I];
    n = size(A,2);
    m = size(A,1);
    kk=1;

    x=zeros(24*n_steps,1500);
    p=zeros(24*n_steps,1500);
    xfinal=zeros(24*n_steps,1);
    U=zeros(12,1);
    x(:,kk) = xinitial;
    z = A*x(:,kk)-b;
```

Appendix - Active Set Code III

```
W = find(abs(z)<=0.0000001);
tW = [1:m];
while kk<500
    Anew = [A([W] ,:)];
    row = size(Anew,1);
    G= Hessian_f(x(:,kk))
    d = grad_f(zeros(size(xinitial)));    % linear part of quadrat
    % function G*xinitial;

    kkt = [G Anew';Anew zeros(row,row)]; %check this also
    %kkt = eye(192);
    g = G*x(:,kk) + d;
    Rhs = [g;zeros(row,1)];
    values = inv(kkt)*Rhs;
    p(:,kk) = -values(1:n,:);
    pk = p(:,kk);
    norm_tol = norm(pk);
    if norm(pk)<0.001
        lambda = values(n+1:length(values),:);
        lambda_=lambda((m_eq+1):size(lambda,1));
        if all(lambda_>=0)
            break
```

Appendix - Active Set Code IV

```
        else
            ConsToRemove=find(lambda==min(lambda));
            for i=ConsToRemove
                if (W(ConsToRemove)>m_eq)
                    W(ConsToRemove) = [];
                end
            end
            x(:,kk+1) = x(:,kk);
            kk = kk+1;
        end
    else
        nw =  setdiff(tW,W);
        alph = [];
        for i=nw
            if A(i,:)*pk < 0
                alph(i) = (b(i) - A(i,:)*x(:,kk))/(A(i,:)*pk);
            end
        end
        alph(alph<=0.00001) = 10;
        alphak = min([1,alph])

        Wnew = find(alph<=(alphak+10^-3));
```

Appendix - Active Set Code V

```
            x(:,kk+1) = x(:,kk) + alphak*pk;
            kk = kk+1;
            x(39:3:end,kk)
            W = [W;Wnew'];
        end
    end

    U=zeros(12,1);
    U(:,1)=x((12*(n_steps)+1):(12*(n_steps)+12),kk-1);

    function [D_,C,D_E,C_E] = equality_constraint()
        C = zeros(12*n_steps,24*n_steps);
        C_E = zeros(12*n_steps,1);
        uConsCount = 1;
        D=zeros(12*n_steps,24*n_steps);
        for j = 1:length(onGround)
            if onGround(j)==0
                D(uConsCount:uConsCount+2,:)=zeros(3,24*n_steps);
                D(uConsCount:uConsCount+2,12*n_steps + 3*j -2: ...
                    12*n_steps + 3*j) = eye(3,3);
                uConsCount = uConsCount+3;
            end
```

Appendix - Active Set Code VI

```
    end
    D_=D(1:(uConsCount-1),:);
    D_E=zeros(size(D_,1),1);
    for i=0:(n_steps-1)
        if i==0
            C(1:12,1:12) = eye(12);
            C(1:12,12*(i)+12*n_steps+1:12*(i)+12*n_steps+12) = ..
            -Bi(1:12,1:12,i+1);
            C_E(1:12) = Ai(:,:,1)*x0+Bi(:,end,i+1);
        else
            C(12*i+1:12*i+12,12*i+1:12*i+12) = eye(12);
            C(12*i+1:12*i+12,12*(i-1)+1:12*(i-1)+12) = ...
            -Ai(1:12,1:12,i+1);
            C(12*i+1:12*i+12,12*i+12*n_steps+1:12*i+12*n_steps+12
            = -Bi(1:12,1:12,i+1);
            C_E(12*i+1:12*i+12)=Bi(:,end,i+1);
        end
    end
end

function [I,b_I] = Inequality_cons()
    I = zeros(24*n_steps,24*n_steps);
```

Appendix - Active Set Code VII

```
eps=0.01;
b_l = zeros(24*n_steps,1);
for i = 0:(n_steps-1)
    for j = 1:4
        if i==0
            l(6*(j-1)+1,12*i+12*n_steps+1+3*(j-1)+2) = 1;
            l(6*(j-1)+2,12*i+12*n_steps+1+3*(j-1)+2) = -1;
            l(6*(j-1)+3,12*i+12*n_steps+1+3*(j-1)+0) = 1 ;
            l(6*(j-1)+3,12*i+12*n_steps+1+3*(j-1)+2) = mu;
            l(6*(j-1)+4,12*i+12*n_steps+1+3*(j-1)+0) = -1;
            l(6*(j-1)+4,12*i+12*n_steps+1+3*(j-1)+2) = mu;
            l(6*(j-1)+5,12*i+12*n_steps+1+3*(j-1)+1) = 1;
            l(6*(j-1)+5,12*i+12*n_steps+1+3*(j-1)+2) = mu;
            l(6*(j-1)+6,12*i+12*n_steps+1+3*(j-1)+1) = -1;
            l(6*(j-1)+6,12*i+12*n_steps+1+3*(j-1)+2) = mu;
        else
            l(24*i+6*(j-1)+1,12*i+12*n_steps+1+3*(j-1)+2) = 1
            l(24*i+6*(j-1)+2,12*i+12*n_steps+1+3*(j-1)+2) = -
            l(24*i+6*(j-1)+3,12*i+12*n_steps+1+3*(j-1)+0) = 1
            l(24*i+6*(j-1)+3,12*i+12*n_steps+1+3*(j-1)+2) = m
            l(24*i+6*(j-1)+4,12*i+12*n_steps+1+3*(j-1)+0) = -
            l(24*i+6*(j-1)+4,12*i+12*n_steps+1+3*(j-1)+2) = m
```

Appendix - Active Set Code VIII

```
                l(24*i+6*(j-1)+5,12*i+12*n_steps+1+3*(j-1)+1) = 1
                l(24*i+6*(j-1)+5,12*i+12*n_steps+1+3*(j-1)+2) = m
                l(24*i+6*(j-1)+6,12*i+12*n_steps+1+3*(j-1)+1) = -
                l(24*i+6*(j-1)+6,12*i+12*n_steps+1+3*(j-1)+2) = m
            end
        end
    end

    for i = 0:n_steps-1
        for j = 1:4
            if i==0
                b_l(6*(j-1)+1) = fmin;
                b_l(6*(j-1)+2) = fmax;
                b_l(6*(j-1)+3) = eps;
                b_l(6*(j-1)+4) = -eps;
                b_l(6*(j-1)+5) = eps;
                b_l(6*(j-1)+6) = -eps;
            else
                b_l(24*i+6*(j-1)+1) = fmin;
                b_l(24*i+6*(j-1)+2) = fmax;
                b_l(24*i+6*(j-1)+3) = eps;
                b_l(24*i+6*(j-1)+4) = -eps;
```

Appendix - Active Set Code IX

```
                    b_l(24*i+6*(j-1)+5) = eps;
                    b_l(24*i+6*(j-1)+6) = -eps;
                end
            end
        end

        l=l(to_keep_constraints,:);
        b_l=b_l(to_keep_constraints,:);
    end

    function del2F=Hessian_f(xstar)
        x_=xstar;
        n_=size(x_,1);
        del2F=zeros(n_,n_);
        h=0.001;
        for i_=1:int8(n_)
            for j_=(double(i_)+1):double(n_)
                e=zeros(n_,1);
                e(i_)=1;
                e(j_)=1;
                A__=main_function(x_+h*e);
                B__=main_function(x_-h*e);
```

Appendix - Active Set Code X

```
                e(i_)=-1;
                C__=main_function(x_+h*e);
                D__=main_function(x_-h*e);
                del2F(i_,j_)=(A__+B__-C__-D__)/(4*h*h);
                del2F(j_,i_)=del2F(i_,j_);
            end
        end
        for i_=1:n_
            e=zeros(n_,1);
            e(i_)=1;
            del2F(i_,i_)=(main_function(x_+h*e)-2*main_function(x_)+
                main_function(x_-h*e))/(h*h);
        end
    end

    function model = main_function(X)
        model = 0;
        for i_=0:n_steps-1
            if i_==0
                j_=double(1);
                p__ = 12*double(n_steps) +1;
            else
```

Appendix - Active Set Code XI

```
            j_ = double(12*i_+1);
            p__ = double(12*n_steps + 12*(i_) + 1);
        end
        model = model + weighted_norm(X(j_:j_+11) - Xref_(j_:j_+1
        + weighted_norm(X(p__:p__+11),R);
    end
end

function N = weighted_norm(a,weight)
    N = a'*weight*a;
end

function delF=grad_f(xstar)
    n_ = length(xstar);     % upto which the definite loop will ru
    h = 0.001;
    delF=zeros(n_,1);
    for i_ = 1:n_
        e = zeros(n_,1);
        e(i_,1) = 1;     % ei vector for ith component
        xstar_plus_ei_h = xstar+e*h;
        xstar_minus_ei_h = xstar-e*h;
        diff_f_wrt_xi = (main_function(xstar_plus_ei_h)- ...
```

Appendix - Active Set Code XII

```
                main_function(xstar_minus_ei_h))/(2*h);
            % using central difference
            delF(i_,1) = diff_f_wrt_xi;
        end
    end
end
```

Appendix - $\mathrm{A_{qp}}$, $\mathrm{B_{qp}}$ code I

```
function [Aqp,Bqp] = Aqp_Bqp(X_ref,n_steps,dt,m,g,I,r,onGround)
    tempA=eye(13,13);
    Aqp=[];
    Bqp=[];
    for k=1:n_steps
        R=[cos(X_ref(3,k)), sin(X_ref(3,k)), 0;
          -sin(X_ref(3,k)), cos(X_ref(3,k)), 0;
                         0,               0, 1];

       I_=(R*I*R')^-1;

        A=[zeros(3), zeros(3),        R, zeros(3), zeros(3,1);
           zeros(3), zeros(3), zeros(3),   eye(3), zeros(3,1);
           zeros(3), zeros(3), zeros(3), zeros(3), zeros(3,1);
           zeros(3), zeros(3), zeros(3), zeros(3),          g;
           zeros(1,3),zeros(1,3),zeros(1,3),zeros(1,3),     0];
        B=[];
        UCount=0;
        for leg=1:4
            if(onGround(leg,k)==1)
                B=[B,[ zeros(3);zeros(3);I_*CP(r(:,leg,k));eye(3)./m;
                UCount=UCount+3;
```

Appendix - $\mathrm{A_{qp}}$, $\mathrm{B_{qp}}$ code II

```
                end
            end
        Ai=exp(A.*dt);
        Bi=integral(@(t) exp(A.*t)*B,0,dt,"ArrayValued",true);
        tempA=Ai*tempA;
        Aqp=[Aqp;tempA];
        if(k==1)
            Bqp=Bi;
        else
            Bqp=[Bqp,zeros((k-1)*13,UCount)];
            Bqp=[Bqp;Ai*Bqp(end-12:end,:)];
            Bqp(end-12:end,(end-UCount+1):end)=Bi;
        end
    end
    function R = CP(r)
        R=[0 -r(3) r(2);r(3) 0 -r(1);-r(2) r(1) 0];
    end
end
```

Appendix - Active Set by QP Code I

```
function [U,Xf] = Active_set_QP(x0,Xref,n_steps,mu,fmin,fmax, ...
    Q_,R_,onGround,Bqp,Aqp)

%initial guess
Xref_=reshape(Xref,1,[])';
onGround=reshape(onGround,1,[])';
R=zeros(size(Bqp,2),size(Bqp,2));
legOnGroundCount=size(Bqp,2)/3;
for i=1:legOnGroundCount
    R(((i-1)*3+1):3*i,((i-1)*3+1):3*i)=R_;
end
Q=zeros(13*n_steps,13*n_steps);
for i=1:n_steps
    Q(((i-1)*13+1):13*i,((i-1)*13+1):13*i)=Q_;
end
xinitial=zeros(size(Bqp,2),1);
xinitial(3:3:end)=(fmin+fmax)*0.5;
m_eq=0;
[I,b_I] = Inequality_cons();
A = I;
b = b_I;
n = size(A,2);
```

Appendix - Active Set by QP Code II

```
m = size(A,1);
kk=1;

G=(Bqp'*Q*Bqp+R).*2;
d=Bqp'*Q*(Aqp*x0-Xref_).*2;
x(:,kk) = xinitial;
z = A*x(:,kk)-b;
W = find(abs(z)<=0.0000001);
tW = [1:m];

while kk<500
    Anew = [A([W],:)];
    row = size(Anew,1);
    kkt = [G Anew'; Anew zeros(row,row)];
    g = G*x(:,kk) + d;
    Rhs = [g; zeros(row,1)];
    values = inv(kkt)*Rhs;
    p(:,kk) = -values(1:n,:);
    pk = p(:,kk);
    if norm(pk)<0.001
        lambda = values(n+1:length(values),:);
        lambda_=lambda((m_eq+1):size(lambda,1));
```

Appendix - Active Set by QP Code III

```
        if all(lambda_>=0)
            break
        else
            ConsToRemove=find(lambda==min(lambda));
            for i=ConsToRemove
                if (W(ConsToRemove)>m_eq)
                    W(ConsToRemove) = [];
                end
            end
            x(:,kk+1) = x(:,kk);
            kk = kk+1;
        end
    else
        nw = setdiff(tW,W);
        alph = [];
        for i=nw
            if A(i,:)*pk < 0
                alph(i) = (b(i) - A(i,:)*x(:,kk))/(A(i,:)*pk);
            end
        end
        alph(alph<=0.00001) = 10;
        alphak = min([1,alph]);
```

Appendix - Active Set by QP Code IV

```
            Wnew = find(alph<=(alphak+10^-3));
            x(:,kk+1) = x(:,kk) + alphak*pk;
            kk = kk+1;
            W = [W;Wnew'];
        end
end
xfinal=x(:,kk);
U=zeros(12,1);
count=1;
for i=1:4
    if(onGround(i)==1)
        U(((i-1)*3+1):i*3)=xfinal(count:count+2);
        count=count+3;
    end
end
Xf=x;

    function [I,b_I] = Inequality_cons()
        I = zeros(2*size(Bqp,2),size(Bqp,2));
        b_I = zeros(size(I,1),1);
        legOnGroundCount=size(Bqp,2)/3;
```

Appendix - Active Set by QP Code V

```
        eps=0.01;
        for i = 1:legOnGroundCount
            l(6*(i-1)+1,(i-1)*3+3) = 1;
            l(6*(i-1)+2,(i-1)*3+3) = -1;
            l(6*(i-1)+3,(i-1)*3+1) = 1  ;
            l(6*(i-1)+3,(i-1)*3+3) = mu;
            l(6*(i-1)+4,(i-1)*3+1) = -1;
            l(6*(i-1)+4,(i-1)*3+3) = mu;
            l(6*(i-1)+5,(i-1)*3+2) = 1;
            l(6*(i-1)+5,(i-1)*3+3) = mu;
            l(6*(i-1)+6,(i-1)*3+2) = -1;
            l(6*(i-1)+6,(i-1)*3+3) = mu;
            b_l(6*(i-1)+1) = fmin;
            b_l(6*(i-1)+2) = fmax;
            b_l(6*(i-1)+3) = eps;
            b_l(6*(i-1)+4) = -eps;
            b_l(6*(i-1)+5) = eps;
            b_l(6*(i-1)+6) = -eps;

        end
    end
end
```

Appendix - Active Set by QP Code VI