Unified Multi-Rate Model Predictive Control for a Jet-Powered Humanoid Robot

Davide Gorbani[1,2], Giuseppe L'Erario[1], Hosameldin Awadalla Omer Mohamed[1], Daniele Pucci[1,2]

Abstract— We propose a novel Model Predictive Control (MPC) framework for a jet-powered flying humanoid robot. The controller is based on a linearised centroidal momentum model to represent the flight dynamics, augmented with a second-order nonlinear model to explicitly account for the slow and nonlinear dynamics of jet propulsion. A key contribution is the introduction of a multi-rate MPC formulation that handles the different actuation rates of the robot's joints and jet engines while embedding the jet dynamics directly into the predictive model. We validated the framework using the jet-powered humanoid robot iRonCub, performing simulations in Mujoco; the simulation results demonstrate the robot's ability to recover from external disturbances and perform stable, non-abrupt flight manoeuvres, validating the effectiveness of the proposed approach.

Fig. 1. iRonCub robot in Mujoco simulator; the thrust forces provided by the jets are visualised as red cylinders.

I. INTRODUCTION

Humanoid robots have traditionally been designed for terrestrial locomotion, mimicking human-like walking and manipulation. However, the advent of flying humanoid robots introduces a fascinating new paradigm — robots that not only resemble human morphology but also possess the ability to fly. This hybrid capability opens up a range of novel applications in disaster response, search and rescue, and exploration, where both agile ground interaction and aerial mobility are essential.

In literature, there are different examples of aerial robots equipped with manipulation or locomotion capabilities [1]; for instance, aerial manipulators consist of a flying platform equipped with a robotic arm or gripper [2], [3].While aerial manipulators are a common example of flying multibody robots, they are not the only type. Other examples include hexapod–quadrotor hybrids [4], ground-and-air capable quadrotors [5], insect-inspired biobots capable of both flying and walking [6], shape-shifting robots that adapt their morphology for different locomotion modes [7], [8], and humanoid robots equipped with thrusters to support bipedal motion [9], [10]. An effort to unify manipulation, aerial mobility, and terrestrial locomotion on a single platform is being carried out by the iRonCub project, which aims to enable the humanoid robot iCub 3 [11] to fly by adding four jet engines on its arms and shoulders.

Controllers for aerial vehicles have been extensively studied, with strategies ranging from classical PID control to advanced nonlinear and optimal control techniques and data-driven techniques [12], [13], [14]. Other examples of control techniques used for aerial systems are controllers that exploit the *thrust-vectoring* for quadrotors with tilting propellers [15].

The above-mentioned controllers are not designed for articulated robots but for single-body aerial systems, such as quadcopters; previous research on jet-powered humanoid control has primarily focused on momentum-based strategies. [16] proposes a framework for asymptotically stabilising the centroidal momentum of the iCub robot, employing a vectored-thrust approach within a hierarchical control structure. Building upon this, [17] introduced a task-based controller capable of tracking full position and attitude trajectories, addressing prior limitations in angular momentum handling and improving flight orientation control. [18] proposes a nonlinear MPC to stabilise both legged and aerial phases of a trajectory performed by the robot.

However, all of these controllers neglected the dynamics of the jet engines, whose behaviour is highly nonlinear, as highlighted in [19] and [20], where the jets were modelled as second-order nonlinear systems. To incorporate jet dynamics in flight control strategies, [21] proposed an approach for trajectory generation; nevertheless, this method remains limited by its offline nature and reliance on a whole-body controller for stabilisation, rendering it unsuitable for real-time control.

A more recent solution was proposed in [22], where a Reinforcement Learning framework was developed to enable the robot to walk and fly, explicitly accounting for jet dynamics. However, this framework requires precise torque-level control, which is not available on all robotic platforms, and it neglects the limitations posed by the low bandwidth actuation of the jets.

To address the challenge of designing an online flight controller that accounts for the complex dynamics of jet propulsion, we propose a Model Predictive Control (MPC) framework that integrates the centroidal momentum dynamics of the robot [23] with the second-order nonlinear jet model introduced in [19]. To ensure compatibility with

[1]Artificial and Mechanical Intelligence, Istituto Italiano di Tecnologia, Genoa, Italy `firstname.surname@iit.it`
[2]School of Computer Science, University of Manchester, Manchester, UK

744

high-frequency online requirements, both the centroidal and jet dynamics are linearised, resulting in a Linear Parameter-Varying (LPV) MPC formulation.

Moreover, to manage the differing control frequencies of the robot's joints and jet engines, we introduce a strategy that allows the MPC to compute control inputs at distinct rates for each actuator type. While previous frameworks have addressed multi-rate systems using hierarchical or cascaded control architectures [24], [25], or have focused on systems with mismatched sampling and actuation periods [26], our approach proposes a unified MPC framework that natively generates control commands at multiple frequencies without relying on hierarchical decomposition.

The remainder of the paper is organised as follows: Section II introduces the notation and reminds the *floating base* formalism; Section III explains the mathematical formulation of the proposed MPC; Section IV shows the results achieved in simulation on the flying humanoid robot iRonCub; Section V concludes the paper with limitations and future directions.

II. BACKGROUND

A. Notation

- $\mathcal{J}$ denotes the world frame;
- $\mathcal{B}$ denotes the body frame;
- $\mathcal{G}[\mathcal{B}]$ denotes a frame with the origin in the centre of mass and the orientation of the body frame;
- $\mathcal{G}[\mathcal{J}]$ denotes a frame with the origin in the centre of mass and the orientation of the inertial frame;
- $^{\mathcal{A}}R_{\mathcal{B}}$ is the rotation matrix from frame $\mathcal{A}$ to frame $\mathcal{B}$;
- $^{\mathcal{A}}p_{\mathcal{B},\mathcal{C}} \in \mathbb{R}^3$ denotes the vector connecting the origin of the frame $\mathcal{B}$ to the origin of frame $\mathcal{C}$ expressed in frame $\mathcal{A}$;
- $^{\mathcal{A}}\omega_{\mathcal{B}}$ is the angular velocity of frame $\mathcal{B}$ respect to the inertial frame $\mathcal{J}$ expressed in frame $\mathcal{A}$;
- $S(x) \in \mathbb{R}^{3\times3}$ is the skew-symmetric matrix such that $S(x)y = x \times y$, where $\times$ is the cross product operator in $\mathbb{R}^3$;

B. Floating-base model

A flying humanoid robot can be modelled as a multi-body system composed of $n+1$ rigid bodies, called links, connected by n joints with one degree of freedom. Following the *floating base* formalism, it is possible to define the configuration space $\mathbb{Q} \in \mathbb{R}^3 \times SO(3) \times \mathbb{R}^n$; an element of the space is composed of the tuple $q = (^{\mathcal{J}}p_{\mathcal{J},\mathcal{B}}, {}^{\mathcal{J}}R_{\mathcal{B}}, s)$. The velocity belongs to the space $\mathbb{V} \in \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}^n$ and an element of the space $\mathbb{V}$ is $\nu = (^{\mathcal{J}}\dot{p}_{\mathcal{J},\mathcal{B}}, {}^{\mathcal{J}}\omega_{\mathcal{B}}, \dot{s})$. By applying the Euler-Poincaré formalism, the equation of motion for a multi-body system subject to m external forces becomes [27]:

$$M(q)\dot{\nu} + C(q,\nu)\nu + G(q) = \begin{bmatrix} 0_{6\times1} \\ \tau \end{bmatrix} + \sum_{k=1}^{m} J_k^\top F_k, \quad (1)$$

where the matrices $M, C \in \mathbb{R}^{(n+6)\times(n+6)}$ are the mass and the Coriolis matrix, the term $G \in \mathbb{R}^{n+6}$ is the gravity vector, the term $\tau \in \mathbb{R}^n$ are the internal actuation torques; F_k is the is the kth of m external forces applied on a frame

$\mathcal{C}_k$ attached to the robot and J_k is the Jacobian that maps the velocity of the robot ν into the velocity of the frame $^{\mathcal{J}}\dot{p}_{\mathcal{C}_k}$ of the origin of $\mathcal{C}_k$.

III. METHOD

A. Centroidal momentum equations

To formulate the MPC, we chose to use the centroidal momentum equation rather than the equation of motion in (1). This approach models the robot's dynamics with fewer variables, leading to a smaller optimisation problem that demands less computational effort and time. As a result, the proposed controller becomes more suitable for real-time implementation on the robot. The *centroidal momentum* equation is [23]:

$$^{\mathcal{G}[\mathcal{J}]}h = \begin{bmatrix} ^{\mathcal{G}[\mathcal{J}]}h^p \\ ^{\mathcal{G}[\mathcal{J}]}h^w \end{bmatrix} = \begin{bmatrix} m^{\mathcal{J}}v_{\mathcal{J},\mathcal{G}} \\ ^{\mathcal{G}[\mathcal{J}]}\mathbb{I}^{\mathcal{J}}\omega_o \end{bmatrix} \in \mathbb{R}^6, \quad (2)$$

where m is the mass of the robot, $^{\mathcal{G}[\mathcal{J}]}h^p$ is the linear momentum, $^{\mathcal{G}[\mathcal{J}]}h^w$ is the angular momentum, $^{\mathcal{G}[\mathcal{J}]}\mathbb{I}(q) \in \mathbb{R}^{3\times3}$ is the robot total inertia expressed in the frame $\mathcal{G}[\mathcal{J}]$ and $^{\mathcal{J}}\omega_o$ is the so called *locked* (or average) angular velocity [23]. As proposed in [17], premultiplying (2) by the rotation matrix $^{\mathcal{B}}R_{\mathcal{J}}$, we obtain a change of coordinate in $\mathcal{G}[\mathcal{B}]$:

$$^{\mathcal{G}[\mathcal{B}]}h = \begin{bmatrix} ^{\mathcal{B}}R_{\mathcal{J}}{}^{\mathcal{G}[\mathcal{J}]}h^p \\ ^{\mathcal{B}}R_{\mathcal{J}}{}^{\mathcal{G}[\mathcal{J}]}h^w \end{bmatrix} = \begin{bmatrix} m^{\mathcal{B}}v_{\mathcal{J},\mathcal{G}} \\ ^{\mathcal{G}[\mathcal{B}]}\mathbb{I}^{\mathcal{B}}\omega_o \end{bmatrix}, \quad (3)$$

where $^{\mathcal{G}[\mathcal{B}]}\mathbb{I} = {}^{\mathcal{B}}R_{\mathcal{J}}{}^{\mathcal{G}[\mathcal{J}]}\mathbb{I}^{\mathcal{J}}R_{\mathcal{B}}$ is the total inertia in new coordinates.

The time derivative of (3) is:

$$^{\mathcal{G}[\mathcal{B}]}\dot{h} = \begin{bmatrix} ^{\mathcal{B}}\dot{R}_{\mathcal{J}}{}^{\mathcal{G}[\mathcal{J}]}h^p + {}^{\mathcal{B}}R_{\mathcal{J}}{}^{\mathcal{G}[\mathcal{J}]}\dot{h}^p \\ ^{\mathcal{B}}\dot{R}_{\mathcal{J}}{}^{\mathcal{G}[\mathcal{J}]}h^w + {}^{\mathcal{B}}R_{\mathcal{J}}{}^{\mathcal{G}[\mathcal{J}]}\dot{h}^w \end{bmatrix}, \quad (4)$$

The terms $^{\mathcal{B}}R_{\mathcal{J}}{}^{\mathcal{G}[\mathcal{J}]}\dot{h}^p$ and $^{\mathcal{B}}R_{\mathcal{J}}{}^{\mathcal{G}[\mathcal{J}]}\dot{h}^w$ are defined as:

$$^{\mathcal{B}}R_{\mathcal{J}}{}^{\mathcal{G}[\mathcal{J}]}\dot{h}^p = A_{lin}(s)T + mg^{\mathcal{B}}R_{\mathcal{J}}e_3, \quad (5)$$

$$^{\mathcal{B}}R_{\mathcal{J}}{}^{\mathcal{G}[\mathcal{J}]}\dot{h}^w = A_{ang}(s)T, \quad (6)$$

where s represents the joint positions and T is the vector collecting the thrust forces $T = [T_1, \ldots, T_{n_j}]$ of the n_j jet turbines, the matrix A is defined as:

$$A_{lin}(s) = \begin{bmatrix} ^{\mathcal{B}}R_{J_1}e_3 & \ldots & ^{\mathcal{B}}R_{J_4}e_3 \end{bmatrix}, \quad (7)$$

$$A_{ang}(s) = \begin{bmatrix} S(^{\mathcal{B}}p_{J_1,\mathcal{G}})^{\mathcal{B}}R_{J_1}e_3 \ldots S(^{\mathcal{B}}p_{J_4,\mathcal{G}})^{\mathcal{B}}R_{J_4}e_3, \end{bmatrix} \quad (8)$$

and the vector e_3 is:

$$e_3 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^\top.$$

Exploiting the definition of the derivative of the rotation matrix $^{\mathcal{B}}\dot{R}_{\mathcal{J}} = -S(^{\mathcal{B}}\omega_{\mathcal{B}})^{\mathcal{B}}R_{\mathcal{J}}$ and combining (4), (5), and (6), we get:

$$^{\mathcal{G}[\mathcal{B}]}\dot{h} = \begin{bmatrix} -S(^{\mathcal{B}}\omega_{\mathcal{B}})^{\mathcal{G}[\mathcal{B}]}h^p + A_{lin}(s)T + mg^{\mathcal{B}}R_{\mathcal{J}}e_3 \\ -S(^{\mathcal{B}}\omega_{\mathcal{B}})^{\mathcal{G}[\mathcal{B}]}h^w + A_{ang}(s)T \end{bmatrix}. \quad (9)$$

Expressing (2) in the $\mathcal{G}[\mathcal{B}]$ coordinates enables the matrices $A_{lin}(s)$ and $A_{ang}(s)$ to be represented exclusively as functions of the joints' internal configuration s. Specifically, the matrices depend on the rotation matrices between the base

frame and each turbine frame, denoted as $^{\mathcal{B}}R_{J_i}, i \in 1, \ldots, 4$, and the relative position vectors between the centre of mass and each turbine frame, $^{\mathcal{B}}p_{J_i,G}, i \in 1, \ldots, 4$. Since these quantities are determined solely by the joints' configuration, this formulation facilitates the subsequent linearization of the matrices, as detailed in Section III-C.

Assumption 1: To simplify the equations of motion, we approximate the angular component of the centroidal momentum as:

$$^{\mathcal{G}[\mathcal{B}]}h^w \approx {}^{\mathcal{G}[\mathcal{B}]}\mathbb{I}\,{}^{\mathcal{B}}\omega_{\mathcal{B}}, \tag{10}$$

under the assumption that $^{\mathcal{B}}\omega_{\mathcal{G}} \approx {}^{\mathcal{B}}\omega_{\mathcal{B}}$. This approximation holds when the joint velocities are sufficiently small, a condition that is satisfied during flight, as the robot moves its arms slowly to adjust the jet directions [17]. With this assumption, the relationship between the derivative of the Euler angles $\phi \in \mathbb{R}^3$ and the angular momentum $^{\mathcal{G}[\mathcal{B}]}h^w$ can be expressed as:

$$\dot{\phi} = ({}^{\mathcal{G}[\mathcal{B}]}\mathbb{I}W)^{-1}\,{}^{\mathcal{G}[\mathcal{B}]}h^w, \tag{11}$$

where W denotes the transformation matrix mapping the body-frame angular velocity $^{\mathcal{B}}\omega_{\mathcal{B}}$ to the Euler angle rates $\dot{\phi}$ [28].

B. Dynamical model

To model the jet dynamics, we adopt the nonlinear second-order model proposed in [19], expressed as:

$$\ddot{T}_i = h(T_i, \dot{T}_i) + g(T_i, \dot{T}_i)v(u_{th,i}), \tag{12}$$

where T_i is the thrust of the ith jet turbines, $u_{th,i}$ represents the throttle of the ith jet turbines, and $v(u_{th,i})$ is a function designed to be invertible with respect to $u_{th,i}$. For notational simplicity, we will refer to $v(u_{th,i})$ simply as v_i.

By incorporating this jet model, the state vector z describing the dynamics of the robot is defined as:

$$z = \begin{bmatrix} ^{\mathcal{J}}x_{\mathcal{G}} & ^{\mathcal{G}[\mathcal{B}]}h^p & \phi & ^{\mathcal{G}[\mathcal{B}]}h^w & T & \dot{T} \end{bmatrix}^\top, \tag{13}$$

where $^{\mathcal{J}}x_{\mathcal{G}}$ denotes the centre of mass position, ϕ represents the Euler angles, $^{\mathcal{G}[\mathcal{B}]}h^p$ and $^{\mathcal{G}[\mathcal{B}]}h^w$ are the linear and angular momentum described in Subsection III-A, and T and $\dot{T}$ denote the thrust and its rate of change, respectively. The control input vector u is given by:

$$u = \begin{bmatrix} s & v \end{bmatrix}^\top, \tag{14}$$

where s represents the joint positions, and v is the vector collecting the v_i auxiliary input related to the throttle command applied to the ith jet engines.

The system of equations that describe the dynamics of the mechanical system is as follows:

$$\begin{cases} ^{\mathcal{J}}\dot{x}_{\mathcal{G}} = \frac{1}{m}{}^{\mathcal{J}}R_{\mathcal{B}}\,{}^{\mathcal{G}[\mathcal{B}]}h^p \\ ^{\mathcal{G}[\mathcal{B}]}\dot{h}^p = A_{lin}(s)T + mg^{\mathcal{B}}R_{\mathcal{J}}e_3 - S(^{\mathcal{B}}\omega_{\mathcal{B}})^{\mathcal{G}[\mathcal{B}]}h^p \\ \dot{\phi} = ({}^{\mathcal{G}[\mathcal{B}]}\mathbb{I}W)^{-1}{}^{\mathcal{B}}R_{\mathcal{J}}\,{}^{\mathcal{G}[\mathcal{B}]}h^w \\ ^{\mathcal{G}[\mathcal{B}]}\dot{h}^w = A_{ang}(s)T - S(^{\mathcal{B}}\omega_{\mathcal{B}})^{\mathcal{G}[\mathcal{B}]}h^w \\ \dot{T}_i = \dot{T}_i \\ \ddot{T}_i = h(T_i, \dot{T}_i) + g(T_i, \dot{T}_i)v_i \end{cases} \tag{15}$$

which we can rewrite in a compact form as:

$$\dot{z} = f(z, u). \tag{16}$$

For the sake of clarity, in the following chapter, the superscript $\mathcal{B}$, $\mathcal{J}$ and $\mathcal{G}[\mathcal{B}]$ will be omitted.

C. Linearised robot dynamics

The resulting dynamic model of the robot is nonlinear with respect to the control inputs. However, solving a nonlinear MPC problem would significantly increase the computational burden and reduce the achievable control frequency. To address this issue, we choose to linearise the model, enabling the use of a linear MPC formulation and thus allowing for higher execution frequencies.

Specifically, we adopt a Linear Parameter-Varying (LPV) approach, where the dynamics are linearised around the current state z_c and control input u_c. Using a first-order Taylor series expansion, the nonlinear dynamic equation (16) can be approximated as:

$$f(z, u) \approx f(z_c, u_c) + \left.\frac{\partial f}{\partial z}\right|_{z_c, u_c}(z - z_c) + \left.\frac{\partial f}{\partial u}\right|_{z_c, u_c}(u - u_c).$$

Assumption 2: In the linearization process, we assume that both the rotation matrix $^{\mathcal{J}}R_{\mathcal{B}}$ and the skew-symmetric matrix $S(\omega_{\mathcal{B}})$ remain constant. This assumption is justified by the fact that, over the relatively short prediction horizon of the MPC, their variation is negligible. Additionally, the inertia matrix $^{\mathcal{B}}\mathbb{I}$ is assumed constant, as it depends only on the joint positions, which are expected to change slowly within the horizon.

Under Assumptions 1 and 2, the linearised model is:

$$\begin{cases} \dot{x}_{\mathcal{G}} \approx \frac{1}{m}{}^{\mathcal{J}}R_{\mathcal{B}}h^p \\ \dot{h}^p \approx A_{lin}(s_c)T + mg^{\mathcal{B}}R_{\mathcal{J}}e_3 + \lambda_{lin}(s - s_c) - S(\omega_{\mathcal{B}})h^p \\ \dot{\phi} \approx (\mathbb{I}W)^{-1}h^w \\ \dot{h}^w \approx A_{ang}(s_c)T + \lambda_{ang}(s - s_c) - S(\omega_{\mathcal{B}})h^w \\ \dot{T}_i = \dot{T}_i \\ \ddot{T}_i \approx h(T_{c,i}, \dot{T}_{c,i}) + \left.\frac{\partial(h + gv)}{\partial T}\right|_{T_{c,i}, \dot{T}_{c,i}, v_{c,i}}(T_i - T_{c,i}) \\ \quad + \left.\frac{\partial(h + gv)}{\partial \dot{T}}\right|_{T_{c,i}, \dot{T}_{c,i}, v_{c,i}}(\dot{T}_i - \dot{T}_{c,i}) + g(T_{c,i}, \dot{T}_{c,i})v_i \end{cases} \tag{17}$$

where the matrices λ_{lin} and λ_{ang} are defined as:

$$\lambda_{lin} = \left.\frac{\partial(A_{lin}(s)T)}{\partial s}\right|_{s_c, T_c}, \quad \lambda_{ang} = \left.\frac{\partial(A_{ang}(s)T)}{\partial s}\right|_{s_c, T_c}. \tag{18}$$

Further details regarding the computation of these matrices are provided in Appendix V-A.

The system 17 can be rewritten as:

$$\dot{z} \approx A(z, u)z + B(z, u)u + c(z, u). \tag{19}$$

where $c(z, u)$ represents the bias terms due to the linearisation. For the sake of clarity, in the following chapters we will refer to the matrices $A(z, u)$ and $B(z, u)$ and to the vector $c(z, u)$ evaluated in the current state of the robot as A, B and c.

D. MPC formulation

In order to improve the tracking of a reference in position and orientation, the state z is augmented to include two additional integral error variables e_x and e_ϕ [29]; the effect of adding those variables is to penalise the steady state offset. The augmented state thus becomes:

$$z = \begin{bmatrix} x_{\mathcal{G}} & h^p & \phi & h^w & T & \dot{T} & e_x & e_\phi \end{bmatrix}^\top, \quad (20)$$

whose dynamics is added to the system in (17) is defined as:

$$\begin{cases} \dot{e}_x = (x_{\mathcal{G}} - x_{\mathcal{G},ref}) \\ \dot{e}_\phi = (\phi - \phi_{ref}) \end{cases} \quad (21)$$

The following formulation for the linearised MPC is implemented:

$$\min_{z,u} \quad \|x_{\mathcal{G}} - x_{\mathcal{G},ref}\|_{W_x}^2 + \|h^p - h_{ref}^p\|_{W_{h^p}}^2$$
$$+ \|\phi - \phi_{ref}\|_{W_\phi}^2 + \|h^w - h_{ref}^w\|_{W_{h^w}}^2$$
$$+ \|\Delta u\|_{W_u}^2 + \|e_x\|_{W_{e_x}}^2 + \|e_\phi\|_{W_{e_\phi}}^2$$

subject to:

$$z_{k+1} = z_k + (Az_k + Bu_k + c)\Delta t, \quad \forall k = 1, \dots, N$$
$$z_0 = z(t)$$
$$s_{min} \leq s_k \leq s_{max}, \quad \forall k = 1, \dots, N$$
$$v_{min} \leq v_k \leq v_{max}, \quad \forall k = 1, \dots, N$$
$$(22)$$

Where $x_{\mathcal{G},ref}$, h_{ref}^p, ϕ_{ref}, and h_{ref}^w are the reference of the centre of mass, linear momentum, Euler angles and angular momentum, and W_x, W_{h^p}, W_ϕ, W_{h^w}, W_u, W_{e_x} and W_{e_ϕ} are diagonal matrices that weight the tasks. The Forward Euler method is used to discretise and propagate the continuous dynamical system of the (19).

E. Multi-rate MPC

The considered jet-powered humanoid robot is equipped with two types of actuators: joints and jet engines. These actuators are controlled by low-level controllers operating at different frequencies. To account for the differences in actuator update rates, we modified the MPC formulation so that it generates new control inputs at frequencies that match those of the respective actuators.

Let us assume that the MPC runs at a frequency f_{MPC} and that a given actuator operates at a lower frequency f_s, with $f_{\text{MPC}} > f_s$. Furthermore, suppose that f_{MPC} is an integer multiple of f_s, such that $N_r = \frac{f_{\text{MPC}}}{f_s}$. In this case, the number of control inputs N_{u_s} associated with the actuator within the prediction horizon is chosen as $N_{u_s} = \frac{N_h}{N_r}$, where N_h is the horizon of the MPC, ensuring that the control input timing matches the actuator's update frequency. Figure 2 shows the variation of two different control inputs with two different control frequencies along the prediction horizon of the MPC. The red line represents a control input u_f that is updated at every propagation step of the MPC, while the blue line represents a control input u_s that is updated every N_r steps of the MPC. Moreover, the control input u_s to be applied to the slow-frequency actuator is computed only at MPC iterations
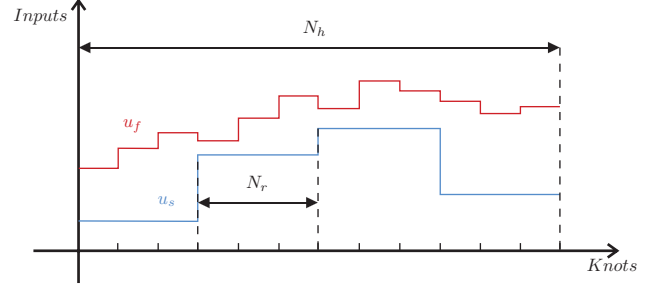

Fig. 2. Multi-rate scheme; the red line represents the control input u_f, which is updated at every MPC step, while the blue line represents the control input u_s, which is updated every N_r steps

satisfying $j = i \cdot N_r$, $i \in \mathbb{N}$. For all other MPC iterations, the first value $u_{s,0}$ is constrained to be equal to the last control input computed for the actuator. Algorithm 1 outlines the multi-rate MPC approach developed to handle actuators operating at different update frequencies. The control inputs for slow actuators are recomputed only at specific MPC iterations aligned with their sampling periods, while previously computed inputs are maintained in the intermediate steps to ensure consistency.

Algorithm 1 Multi-Rate MPC

Require: f_{MPC}, f_s, N_r, u_{prev}, j
1: $N_r \leftarrow f_{\text{MPC}}/f_s$
2: $j \leftarrow N_r$
3: **for** each MPC iteration j **do**
4:     **if** $j = N_r$ **then**
5:         Solve MPC problem
6:         Extract and apply $u_{s,0}$ for slow actuators
7:         $u_{\text{prev}} \leftarrow u_{s,0}$
8:     **else**
9:         Constrain $u_{s,0} = u_{\text{prev}}$ in MPC
10:         Solve MPC problem
11:     **end if**
12:     Extract and apply remaining actuator commands
13: **end for**

IV. RESULTS

In this section, we present the simulation results of the proposed method. We chose to perform a recovery from an external disturbance to prove the robustness of the controller, the tracking of a minimum jerk trajectory, and two ablation studies to determine the impact of the jet dynamics and the multi-rate architecture.

A. Simulation Environment

The simulation environment used to perform the experiments is MuJoCo [30], running at a frequency of 1000 Hz. All simulations are conducted on an Ubuntu 22.04 operating system equipped with an Intel(R) Core(TM) i7-12700H processor[1]. In order to have a simulation that is

---

[1] The code to run the simulations is available at https://github.com/ami-iit/paper_gorbani_2025_humanoids_multi-rate-mpc-ironcub

closer to reality, the jet dynamics are simulated using the neural network model proposed in [22], which is identified using data gathered from the real turbines. To identify the second-order model (12), the *Extended Kalman Filter* (EKF) algorithm described in [19] is employed. Figure 3 presents a comparison between the thrust predicted by the neural network model and the thrust generated by the identified non-linear model, when both are provided with the same throttle input profile. The mean absolute error (MAE) between the two models is approximately 5.95 N. This comparison highlights the mismatch between the model used in simulation and one used in the prediction model of the MPC.

In our case, the low-level controller of the joints operates at 1000 Hz, while the low-level controller of the jet engines operates at 10 Hz. Those values are chosen taking into consideration the specifics of the robot iRonCub. To enable the MPC to run at a higher frequency, we adopt the approach proposed in [31], which adapts the sampling intervals of the MPC along the prediction horizon, with a shorter sampling time at the beginning of the horizon, which progressively increases towards the end. In fact, given the slow dynamics of the jet engines, we chose a control horizon of 1 second. Given that control horizon, to make the MPC run at 200 Hz with a constant timestep, we would need 200 knots, which would considerably slow down the problem. Exploiting this method, we can run the MPC at a frequency of 200 Hz with 17 knots. Recalling the notation in Subsection III-E, the low-frequency control u_s input is the jet turbines control input v, which varies at a frequency of 10 Hz. The joint positions s are computed at every step of the MPC, and the number N_r, since we adopt a variable sampling, varies along the control horizon, but it is chosen such that the jet turbine throttle is kept constant for 0.1 seconds.

B. Simulation results

1) Disturbance recovery: To demonstrate the controller's ability to reject external disturbances, an external torque of 300 Nm was applied along the y-axis, and an external force of 50 N was applied along the x-axis of the root link for 0.1 seconds at $t = 2$ s. As shown in Figure 4, the robot successfully recovers from the disturbance, despite experiencing a tilt of approximately $15°$ in the roll axis, a displacement of about 1 m along the x-axis, and a vertical drop of approximately 1 m.

2) Minimum Jerk Trajectory Tracking: To test the performance of the MPC in tracking a reference trajectory, a minimum jerk trajectory is generated by interpolating a given set of points and used as the reference for the centre of mass position $^{\mathcal{J}}x_{\mathcal{G},ref}$, while the reference orientation ϕ_{ref} is constant. The reference linear momentum is computed as $h^l_{ref} = m^{\mathcal{B}}R_{\mathcal{J}}{}^{\mathcal{J}}\dot{x}_{\mathcal{G},ref}$, while the reference angular momentum h^w_{ref} is set to zero since the reference orientation is constant. Figure 5 shows the tracking of the centre of mass and the orientation of the base. The blue line is the measured signal, while the green one is the reference; the x and y components of the position are tracked fairly well by the controller, while the z component presents an offset of about 0.15 m. Concerning the orientation, on Roll and Yaw have

a Mean Absolute Error lower than 0.01 rad, while for the Pitch it is about 0.03 rad. The error in the tracking is due to the discrepancies between the neural network model used to simulate the jet dynamics and the linearised non-linear model used in the prediction model of the MPC.

C. Real-Time Feasibility

Ensuring real-time execution is critical for online MPC applications on flying robots. In our implementation, the MPC problem is formulated as a quadratic program (QP) and solved using the OSQP solver [32]. As shown in Figure 6, the average computation time per MPC iteration during tracking of the minimum jerk trajectory in Subsubsection IV-B.2 is 2.18 ms with a standard deviation of 0.16 ms and the maximum time of 4.45 ms, which is below the target frequency of 200 Hz (corresponding to a period of 5 ms). Given that the MPC is executed at 200 Hz (5 ms per cycle), the solver's performance is compatible with real-time operation. These results suggest that the proposed control strategy can feasibly run onboard for a complex system like iRonCub, provided that the onboard computational resources are comparable to those used in the simulation setup.

D. Ablation Study

1) Jet Dynamics study: To demonstrate the necessity of embedding the jet dynamics inside the MPC framework, we performed an ablation study where we removed the jet dynamics from the MPC and let the controller choose a thrust at will (subject to a regularisation cost); to compute the control input for the jet turbines, we used the Feedback Linearisation controller proposed in [19]. Testing the minimum jerk trajectory presented in IV-B.2, the controller failed to stabilise the robot, and the robot fell on the ground.

2) LPV study: To demonstrate the relevance of the LPV formulation, we carried out a simulation in which the model was linearised about the robot's initial position and commanded to follow a minimum-jerk trajectory. In this case too, the robot was unable to track the trajectory and eventually fell.

3) Multi-rate study: To isolate the impact of the *multi-rate* scheme, we compare our proposed controller with a *single-rate* MPC variant. In the single-rate setup, the MPC computes jet thrust commands at every iteration (200 Hz), but only the commands at 10 Hz are applied, discarding the intermediate control inputs. This reflects the behaviour of the turbine's low-level controller, which operates at 10 Hz and ignores any commands received within each 0.1-second interval. Both controllers are evaluated on the minimum jerk trajectory described in Section IV-B.2. Figure 5 presents a comparison of the Multi-rate and Single-rate MPCs. While both achieve similar tracking performance in the x and y position components, the Single-rate MPC exhibits significantly greater oscillations in orientation. This observation is supported by the Mean Absolute Error (MAE) of the centre of mass position and base orientation reported in Table I. These results highlight that explicitly accounting for actuator update rates in the MPC formulation leads to significantly improved tracking performance.
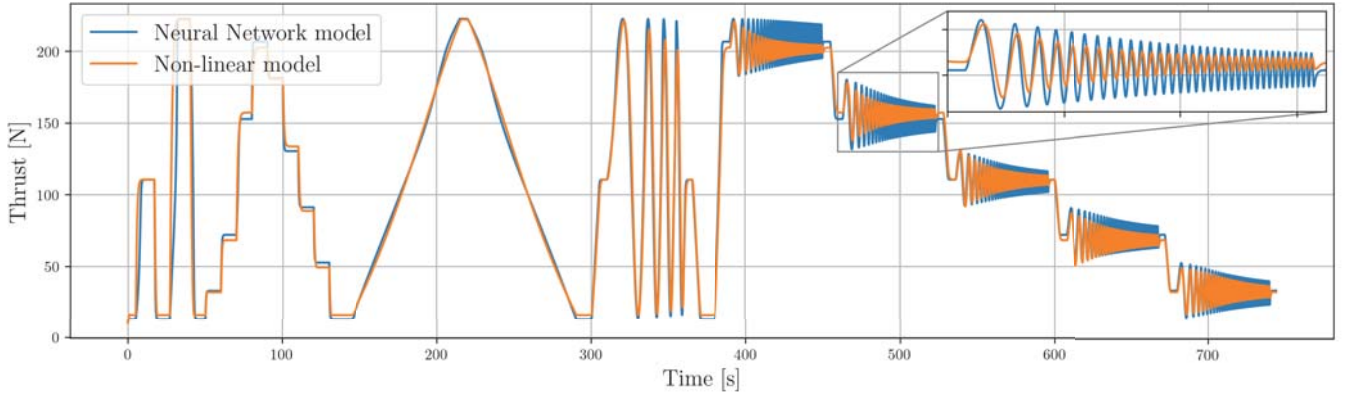
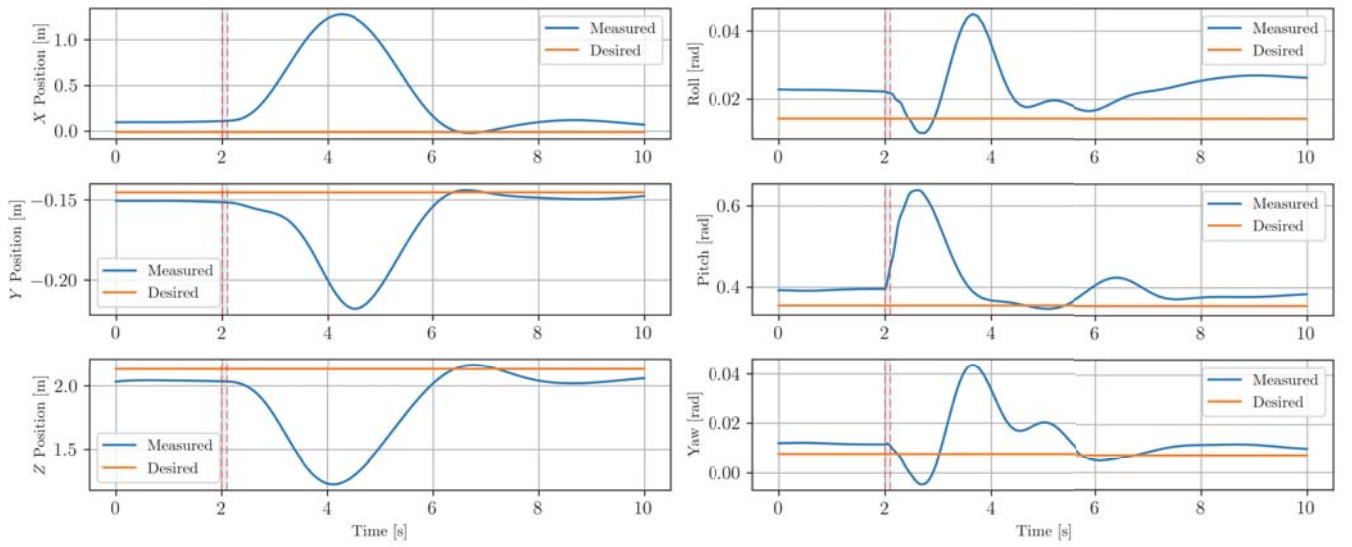Fig. 3. Neural Network and non-linear jet model comparison over the identification signal.

Fig. 4. Centre of mass position and base orientation of the robot subject to an external disturbance applied between 2 s and 2.1 s (corresponding to the shaded region).
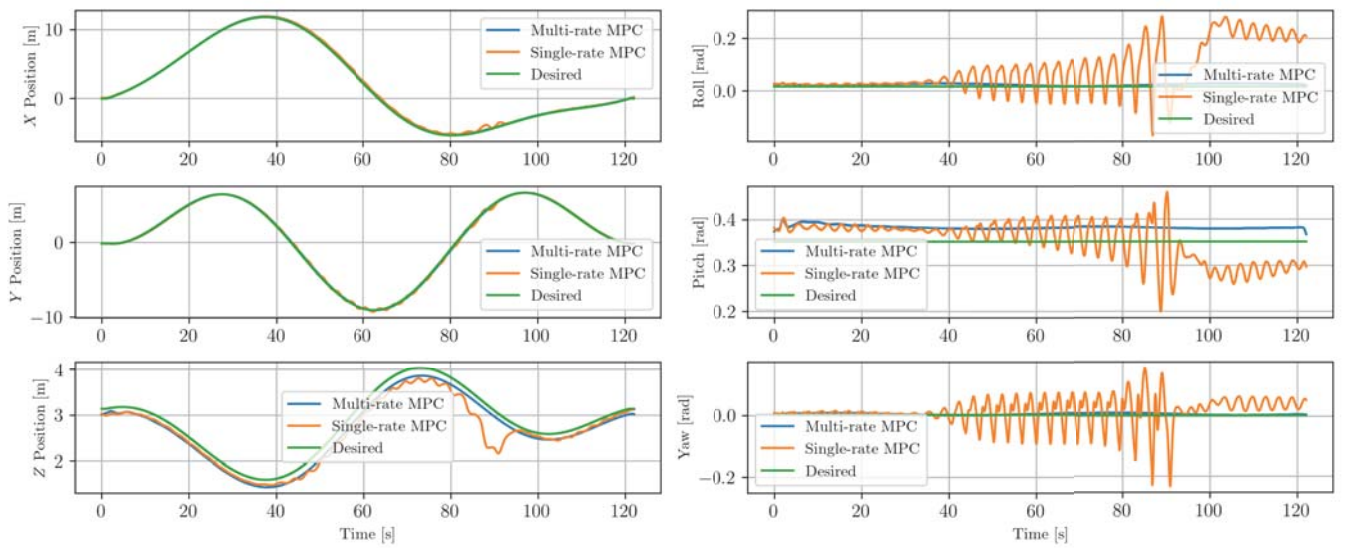

Fig. 5. Comparison between the Multi-rate MPC and the Single-rate MPC in tracking a minimum jerk trajectory.

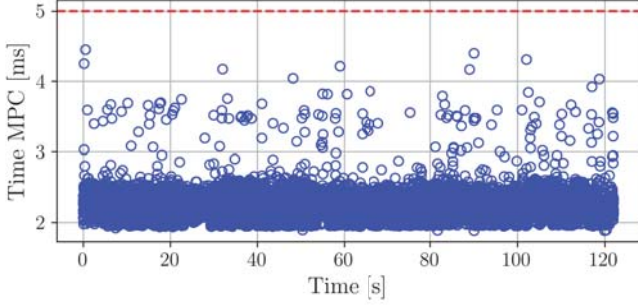|	X	Y	Z	Roll	Pitch	Yaw
Multi-rate	0.1106 m	0.0729 m	0.1508 m	0.0076 rad	0.0307 rad	0.0036 rad
Single-rate	0.1429 m	0.1005 m	0.2034 m	0.0765 rad	0.0371 rad	0.0319 rad

Fig. 6.   Time elapsed to solve each iteration of the MPC.

E. Limitations

One of the main limitations of the proposed MPC formulation is the reliance on Euler angles, which are inherently subject to singularities. However, given our choice of the reference frame in the case of the iRonCub robot, the singularity occurs only when the robot experiences a $90°$ pitch rotation, i.e., when it becomes fully horizontal. Since our platform is designed for VTOL and avoids such horizontal flight, this theoretical limitation doesn't affect its intended operation.

A second limitation arises from the linearisation process, as stated in Assumption 2, where the rotation matrix ${}^B R_{\mathcal{J}}$ and the angular velocity $\omega_{\mathcal{B}}$ are assumed to be constant during the prediction horizon to maintain a linear problem structure. Given that the MPC horizon is set to 1 second and that the robot is not expected to perform aggressive manoeuvres within this time frame, it is reasonable to assume that the orientation and angular velocity remain sufficiently close to the initial condition, thereby justifying this simplification.

V. CONCLUSIONS

In this paper, a momentum-based linearised Model Predictive Control (MPC) is proposed, which accounts for both the dynamics of the jet engines and the limitations in the control frequency of the jets. The controller demonstrates the ability to recover from external disturbances and track a smooth trajectory, even embedding in the MPC an approximated model of the jet dynamics with respect to the simulated one. Simulation-based validation is performed using the jet-powered humanoid robot iRonCub.

The proposed controller is a step forward in the realisation of a flight controller that works on the real robot, since it overcomes two of the limitations related to the hardware, which are the slow dynamics of the jet engines and the low frequency at which the turbines run.

However, several assumptions were made in the formulation of the controller. For example, the simplification of the robot's dynamics and the use of the Euler angles limit the applicability

of this controller for aggressive manoeuvres. Additionally, aerodynamic forces are neglected in the current model.

Future work will involve testing the proposed controller on the real robot, as well as developing a specific controller capable of handling the take-off and landing phases.

APPENDIX

A. Linearization

In this subsection, the computations needed to get the matrices in (18) are presented.

It is possible to compute the time derivative of the term $A_{lin}(s)T$ using the chain rule:

$$\frac{\partial A_{lin}(s)T}{\partial t} = \frac{\partial A_{lin}(s)T}{\partial s}\dot{s} + \frac{\partial A_{lin}(s)T}{\partial T}\dot{T}. \quad (23)$$

The derivative can also be written by exploiting the definition of the derivative of a product:

$$\frac{dA_{lin}(s)T}{dt} = \dot{A}_{lin}(s,\dot{s})T + A_{lin}(s)\dot{T}. \quad (24)$$

Given that the (23) and (24) are equal and since $\frac{\partial A_{lin}(s)T}{\partial T}\dot{T} = A(s)\dot{T}$, it holds that $\frac{\partial A_{lin}(s)T}{\partial s}\dot{s} = \dot{A}(s,\dot{s})T$. Assuming that the number of jets n_j is 4, the matrix $\dot{A}(s,\dot{s})$ is ${}^B\dot{A} = \begin{bmatrix} {}^B\dot{R}_{J_1}e_3 & ... & {}^B\dot{R}_{J_4}e_3 \end{bmatrix}$; exploiting the definition of the derivative of the rotation matrix we can rewrite the derivative as:

$${}^B\dot{A} = \begin{bmatrix} S({}^B\omega_{B,J_1}){}^B R_{J_1}e_3 & ... & S({}^B\omega_{B,J_{n_j}}){}^B R_{J_4}e_3 \end{bmatrix}.$$

Exploiting the property $S(x)y = -S(y)x$ and writing ${}^B\omega_{B,J_j} = J_{\omega_j}^{rel}(s)\dot{s}$ we can finally rewrite:

$${}^B\dot{A} = \begin{bmatrix} -S({}^B R_{J_1}e_3)J_{\omega_1}^{rel}\dot{s} & ... & -S({}^B R_{J_{n_4}}e_3)J_{\omega_{n_j}}^{rel}\dot{s} \end{bmatrix},$$

and therefore it is possible to rearrange ${}^B\dot{A}(s,\dot{s})T$ as:

$${}^B\dot{A}T = \begin{bmatrix} -T_1 S({}^B R_{J_1}e_3) & ... & -T_{n_4}S({}^B R_{J_{n_4}}e_3) \end{bmatrix} \bar{J}_\omega^{rel}\dot{s},$$

where $\bar{J}_\omega^{rel}$ is a matrix composed by the Jacobians of the four jets defined as $\bar{J}_\omega^{rel} = \begin{bmatrix} J_{\omega_1}^{rel,\top} & ... & J_{\omega_4}^{rel,\top} \end{bmatrix}^\top \in \mathbb{R}^{12 \times n_s}$, leading to the definition of $\lambda_{lin}(s,T)$ as:

$$\lambda(s,T) = \begin{bmatrix} -T_1 S({}^B R_{J_1}e_3) & ... & -T_{n_j}S({}^B R_{J_{n_j}}e_3) \end{bmatrix} \bar{J}_\omega^{rel}.$$

The matrix $\lambda_{ang}(s,T)$ can be found in an analogous way starting from the time derivative of the term $A_{ang}(s)T$, which by definition can be written as:

$$\frac{dA_{lin}(s)T}{dt} = \dot{A}_{ang}(s,\dot{s})T + A_{ang}(s)\dot{T}, \quad (25)$$

exploiting the chain rule, the derivative can also be written as:

$$\frac{dA_{ang}(s)T}{dt} = \frac{\partial A_{ang}(s)T}{\partial s}\dot{s} + \frac{\partial A_{ang}(s)T}{\partial T}\dot{T}. \quad (26)$$

Since (25) and (26) are equal and $A_{ang}(s)\dot{T} = \frac{\partial A_{ang}(s)T}{\partial T}\dot{T}$, it holds that $\dot{A}_{ang}(s, \dot{s})T = \frac{\partial A_{ang}(s)T}{\partial s}\dot{s}$. The matrix $\dot{A}_{ang}(s, \dot{s})$ is:

$$\dot{A}_{ang}(s, \dot{s}) = \begin{bmatrix} C_1 & C_2 & C_3 & C_4 \end{bmatrix}, \qquad (27)$$

where each column is defined as:

$$C_i = (S(^B\dot{r}_i)^B R_i e_3 + S(^B r_i)^B \dot{R}_i e_3), \ i \in \{1, 2, 3, 4\}.$$

where $^B r_i$ is the distance between the centre of mass and the ith turbine expressed in body frame. Exploiting once again the definition of the derivative of the rotation matrix, the property $S(x)y = -S(y)x$ and using the linear and angular Jacobians $^B\omega_{B,J_j} = J^{rel}_{\omega_j}(s)\dot{s}$ and $^B\omega_{B,J_j} = J^{rel}_{\omega_j}(s)\dot{s}$, (27) rewrites as:

$$\dot{A}_{ang}(s, \dot{s}) = - \begin{bmatrix} \bar{C}_1 & \bar{C}_2 & \bar{C}_3 & \bar{C}_4 \end{bmatrix},$$

where:

$$\bar{C}_i = (S(^B R_i e_3)^B J^{rel}_{\dot{r}_i} + S(^B r_i)S(^B R_i e_3)J^{rel}_{\omega_i})\dot{s}.$$

The term $\dot{A}_{ang}(s, \dot{s})T$ can be rewritten isolating $\dot{s}$ as:

$$\begin{aligned}
\dot{A}_{ang}(s, \dot{s})T = \\
- (T_1(S(^B R_1 e_3)^B J^{rel}_{\dot{r}_1} + S(^B r_1)S(^B R_1 e_3)J^{rel}_{\omega_1}) + \dots \\
+ T_4(S(^B R_4 e_3)^B J^{rel}_{\dot{r}_4} + S(^B r_4)S(^B R_4 e_3)J^{rel}_{\omega_4}))\dot{s} \\
= \Lambda_{ang}(s, T)\dot{s}.
\end{aligned}$$

REFERENCES

[1] Xidong Zhou, Hang Zhong, Hui Zhang, Wei He, Hean Hua, and Yaonan Wang. Current status, challenges, and prospects for new types of aerial robots. *Engineering*, 2024.

[2] Hossein Bonyan Khamseh, Farrokh Janabi-Sharifi, and Abdelkader Abdessameud. Aerial manipulation—a literature survey. *Robotics and Autonomous Systems*, 107:221–235, 2018.

[3] Anibal Ollero, Marco Tognon, Alejandro Suarez, Dongjun Lee, and Antonio Franchi. Past, present, and future of aerial robotic manipulators. *IEEE Transactions on Robotics*, 38(1):626–645, 2021.

[4] M. Pitonyak and F. Sahin. A Novel Hexapod Robot Design with Flight Capability. In *Proceedings of the 12th System of Systems Engineering Conference*, pages 1–6, 2017.

[5] A. Kalantari and M. Spenko. Design and Experimental Validation of HYTAQ, a Hybrid Terrestrial and Aerial Quadrotor. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4445–4450, 2013.

[6] A. Bozkurt, A. Lal, and R. Gilmour. Aerial and Terrestrial Locomotion Control of Lift Assisted Insect Biobots. In *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 2058–2061, 2009.

[7] L. Daler, S. Mintchev, C. Stefanini, and D. Floreano. A Bioinspired Multi-Modal Flying and Walking Robot. *Bioinspiration & Biomimetics*, 10(1):016005, 2015.

[8] L. Daler, J. Lecoeur, P. B. Hählen, and D. Floreano. A Flying Robot with Adaptive Morphology for Multi-Modal Locomotion. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1361–1366, 2013.

[9] K. Kim, P. Spieler, E. Lupu, A. Ramezani, and S.-J. Chung. A Bipedal Walking Robot that Can Fly, Slackline, and Skateboard. *Science Robotics*, 6(59):eabf8136, 2021.

[10] Z. Huang, B. Liu, J. Wei, Q. Lin, J. Ota, and Y. Zhang. Jet-HR1: Two-Dimensional Bipedal Robot Step Over Large Obstacle Based on a Ducted-Fan Propulsion System. In *Proceedings of the IEEE-RAS 17th International Conference on Humanoid Robots*, pages 406–411, 2017.

[11] Stefano Dafarra, Ugo Pattacini, Giulio Romualdi, Lorenzo Rapetti, Riccardo Grieco, Kourosh Darvish, Gianluca Milani, Enrico Valli, Ines Sorrentino, Paolo Maria Viceconte, et al. icub3 avatar system: Enabling remote fully immersive embodiment of humanoid robots. *Science Robotics*, 9(86):eadh3834, 2024.

[12] Zongyu Zuo, Cunjia Liu, Qing-Long Han, and Jiawei Song. Unmanned aerial vehicles: Control methods and future challenges. *IEEE/CAA Journal of Automatica Sinica*, 9(4):601–614, 2022.

[13] Lénaïck Besnard, Yuri B Shtessel, and Brian Landrum. Quadrotor vehicle control via sliding mode controller driven by sliding mode disturbance observer. *Journal of the Franklin Institute*, 349(2):658–684, 2012.

[14] Guillem Torrente, Elia Kaufmann, Philipp Föhn, and Davide Scaramuzza. Data-driven mpc for quadrotors. *IEEE Robotics and Automation Letters*, 6(2):3769–3776, 2021.

[15] Rumit Kumar, Mahathi Bhargavapuri, Aditya M Deshpande, Siddharth Sridhar, Kelly Cohen, and Manish Kumar. Quaternion feedback based autonomous control of a quadcopter uav with thrust vectoring rotors. In *2020 american control conference (acc)*, pages 3828–3833. IEEE, 2020.

[16] Daniele Pucci, Silvio Traversaro, and Francesco Nori. Momentum control of an underactuated flying humanoid robot. *IEEE Robotics and Automation Letters*, 3(1):195–202, 2017.

[17] Gabriele Nava, Luca Fiorio, Silvio Traversaro, and Daniele Pucci. Position and attitude control of an underactuated flying humanoid robot. In *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*, pages 1–9, 2018.

[18] Saverio Taliani, Gabriele Nava, Giuseppe L'Erario, Mohamed Elobaid, Giulio Romualdi, and Daniele Pucci. Online nonlinear mpc for multimodal locomotion. In *International Conference on Robotics and Automation (ICRA)*. IEEE, 2025.

[19] Giuseppe L'Erario, Luca Fiorio, Gabriele Nava, Fabio Bergonti, Hosameldin Awadalla Omer Mohamed, Emilio Benenati, Silvio Traversaro, and Daniele Pucci. Modeling, identification and control of model jet engines for jet powered robotics. *IEEE Robotics and Automation Letters*, 5(2):2070–2077, 2020.

[20] Affaf Junaid Ahamad Momin, Gabriele Nava, Giuseppe L'Erario, Hosameldin Awadalla Omer Mohamed, Fabio Bergonti, Punith Reddy Vanteddu, Francesco Braghin, and Daniele Pucci. Nonlinear model identification and observer design for thrust estimation of small-scale turbojet engines. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 5879–5885. IEEE, 2022.

[21] Giuseppe L'Erario, Gabriele Nava, Giulio Romualdi, Fabio Bergonti, Valentino Razza, Stefano Dafarra, and Daniele Pucci. Whole-body trajectory optimization for robot multimodal locomotion. In *2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids)*, pages 651–658, 2022.

[22] Giuseppe L'Erario, Drew Hanover, Angel Romero, Yunlong Song, Gabriele Nava, Paolo Maria Viceconte, Daniele Pucci, and Davide Scaramuzza. Learning to walk and fly with adversarial motion priors. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10370–10377. IEEE, 2024.

[23] David E Orin, Ambarish Goswami, and Sung-Hee Lee. Centroidal dynamics of a humanoid robot. *Autonomous robots*, 35:161–176, 2013.

[24] Ugo Rosolia and Aaron D Ames. Multi-rate control design leveraging control barrier functions and model predictive control policies. *IEEE Control Systems Letters*, 5(3):1007–1012, 2020.

[25] Marcello Farina, Xinglong Zhang, and Riccardo Scattolini. A hierarchical multi-rate mpc scheme for interconnected systems. *Automatica*, 90:38–46, 2018.

[26] Keck Voon Ling, WU Bingfang, HE Minghua, and Zhang Yu. A model predictive controller for multirate cascade systems. In *Proceedings of the 2004 American Control Conference*, volume 2, pages 1575–1579. IEEE, 2004.

[27] Silvio Traversaro. Modelling, estimation and identification of humanoid robots dynamics. *Ph. D. dissertation*, 2017.

[28] Teppo Luukkonen. Modelling and control of quadcopter. *Independent research project in applied mathematics, Espoo*, 22(22):1–24, 2011.

[29] Gabriele Pannocchia, Marco Gabiccini, and Alessio Artoni. Offset-free mpc explained: novelties, subtleties, and applications. *IFAC-PapersOnLine*, 48(23):342–351, 2015.

[30] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

[31] Zehui Lu and Shaoshuai Mou. Variable sampling mpc via differentiable time-warping function. In *2023 American Control Conference (ACC)*, pages 533–538, 2023.

[32] Bartolomeo Stellato, Goran Banjac, Paul Goulart, Alberto Bemporad, and Stephen Boyd. Osqp: An operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 12(4):637–672, 2020.