Model Predictive Control for Powered Descent Guidance and Control

Stefano Riccardi, Gaetano Torella, *Unina*

I. Introduction

Future space missions require increasingly higher levels of autonomous guidance and control, especially for critical phases such as Entry, Descent, and Landing (EDL). In the powered descent phase, which represents the final and most delicate segment of EDL, a spacecraft relies on engine thrust to actively regulate its trajectory and ensure a safe and precise touchdown. Pin-point landing capability, defined as the ability to land within a few tens of meters from a target, is considered a key technological challenge and an enabling factor for both planetary landers and reusable launch vehicles [1]. Improving landing precision would significantly enhance the scientific return of planetary exploration missions by allowing spacecraft to reach areas of highest interest, while for launchers it would directly support reusability, reducing costs and increasing mission responsiveness.

Traditional approaches to descent guidance typically rely on pre-computed trajectories combined with relatively simple control laws, which must be designed with wide tolerances to account for uncertainties. However, these strategies often struggle to adapt effectively to disturbances, unforeseen events, or model mismatches that may occur during the descent phase [2]. This limitation is shared with launch operations, where current practices demand extensive pre-launch planning and suffer from strong weather dependencies, leading to scheduling rigidity and higher operational costs.

Recent advances in guidance, navigation, and control (GNC) highlight Model Predictive Control (MPC) as a promising enabling technology for achieving unprecedented levels of autonomy in space applications [1], [3]. MPC offers a systematic framework for multivariable control, capable of explicitly handling the nonlinear and constrained dynamics of a landing vehicle. By leveraging an on-board dynamical model to predict system evolution, MPC continuously solves an optimization problem over a receding horizon, selecting in real time the optimal control actions to mitigate disturbances and adapt to changing conditions [2], [3]. The effectiveness of MPC is strongly supported by recent developments in convex optimization, such as quadratic programming (QP) and second-order cone programming (SOCP).

The present work is devoted to the analysis of a guidance and control strategy for rocket-powered descent based on Model Predictive Control (MPC). The spacecraft model adopted is introduced, the formulation of the MPC problem together with its advantages over classical controllers is outlined, the implementation in MATLAB/YALMIP is described, and simulation results for a representative landing scenario are presented.

II. Spacecraft Model

The rocket lander is modeled as a six-degree-of-freedom (6-DoF) rigid body with thrust vector control.

The aerodynamics, the gravity and the propulsion system generate the forces and the torques about the spacecraft's center of gravity (CoG) that influence its motion. Introducing the inertial and body reference frames as depicted in Fig. 1, where the first one is fixed in space, while the second one is moving linked to the rocket, it is possible to define the equations of motion (EoM) for the translational and rotational dynamics by means of Newton's second law:

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \frac{1}{m} C_I^b \begin{bmatrix} F_{bx} \\ F_{by} \\ F_{bz} \end{bmatrix} + \begin{bmatrix} g_{I_x} \\ g_{I_y} \\ g_{I_z} \end{bmatrix} \tag{1a}$$

$$\begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{xy} & I_{yy} & I_{yz} \\ I_{xz} & I_{yx} & I_{zz} \end{bmatrix}^{-1} \left(\begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} - \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \times \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{xy} & I_{yy} & I_{yz} \\ I_{xz} & I_{yx} & I_{zz} \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \right) \tag{1b}$$

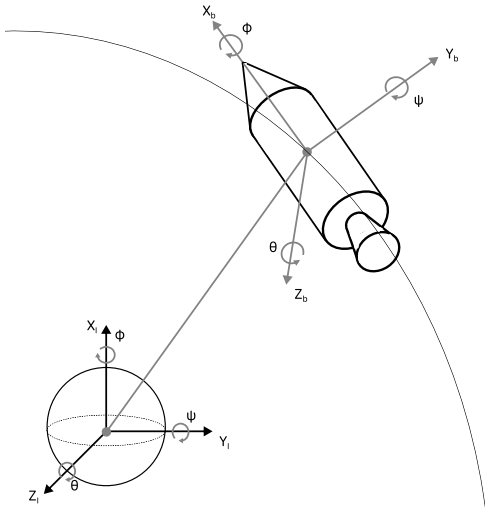

Fig. 1: Inertial $[X_I, Y_I, Z_I]$ and body $[X_b, Y_b, Z_b]$ frames

where m is the vehicle mass, C_I^b is the direction cosine matrix transforming vectors from the body frame to the inertial frame, $\mathbf{F_b} = [F_{bx}, F_{by}, F_{bz}]^T$ is the force vector in the body frame, $\mathbf{g_I} = [g_{I_x}, g_{I_y}, g_{I_z}]^T$ is the gravitational acceleration vector in the inertial frame, $\mathbf{M} = [M_x, M_y, M_z]^T$ is the moment vector about the CoG in the body frame, and I_{xx}, I_{yy}, I_{zz} are the principal moments of inertia with I_{xy}, I_{xz}, I_{yz} being the products of inertia. The vectors $[\ddot{x}, \ddot{y}, \ddot{z}]^T$ and $[\ddot{\phi},\ \ddot{\theta},\ \ddot{\psi}]^T$ represent the translational accelerations and angular accelerations respectively.

For the purposes of controller design, several simplifying assumptions are introduced in order to focus on the dominant dynamics [1]:

- Planetary rotation is neglected, so that the inertial frame can be considered non-rotating during descent.
- The landing surface is assumed flat and the gravity field uniform in both magnitude and direction.
- Aerodynamic effects are considered negligible.
- The inertia matrix is assumed diagonal, with principal axes aligned to the body axes and negligible cross-coupling products.
- The vehicle is actuated through four virtual control inputs: U_1 (rolling moment), U_2 (pitching moment), U_3 (yawing moment), and U_4 (thrust force). Moments are generated by thrust vectoring or auxiliary actuators, while U_4 represents the magnitude of the main engine thrust.

Under these assumptions, the equations of motion can be simplified. Considering an inertial frame $\{x, y, z\}$ (with the x-axis aligned vertically upward and the y, z axes in the horizontal plane) and adopting Euler angles ϕ (roll), θ (pitch), and ψ (yaw) to describe the rocket's orientation, the translational dynamics are expressed as:

$$\ddot{x} = \frac{U_4}{m}\cos\theta\cos\psi - g, \tag{2}$$

$$\ddot{y} = \frac{U_4}{m}\cos\theta\sin\psi, \tag{3}$$

$$\ddot{z} = -\frac{U_4}{m}\sin\theta, \tag{4}$$

where m denotes the vehicle mass, U_4 the thrust force, and g the gravitational acceleration. The term $\frac{U_4}{m}\cos\theta\cos\psi - g$ in the x-direction shows that, when the rocket is upright ($\theta = 0, \psi = 0$) and $U_4 = mg$, the vertical acceleration is zero, corresponding to a hovering condition.

The rotational dynamics are governed by:

$$\ddot{\phi} = \frac{U_1}{I_{xx}}, \quad \ddot{\theta} = \frac{U_2}{I_{yy}}, \quad \ddot{\psi} = \frac{U_3}{I_{zz}}, \tag{5}$$

where I_{xx}, I_{yy}, I_{zz} are the principal moments of inertia. With the diagonal inertia assumption, angular motions about each axis are decoupled. The orientation angles ϕ, θ, ψ determine the attitude of the rocket, and through them the thrust vector is indirectly reoriented to generate the required translational accelerations.

For controller design, the nonlinear dynamics are linearized around a nominal hover condition defined by $\phi = \theta = \psi = 0$ and $U_4 = mg$, corresponding to a stationary hover. Linearization about this equilibrium yields a linear time-invariant (LTI) state-space model (after discretization) that captures the vehicle's small perturbation dynamics. The state vector includes position, velocity, orientation angles, and angular rates (all defined as deviations from the equilibrium), while the control input vector is $\mathbf{u} = [U_1, U_2, U_3, U_4]^T$. This linearized 6-DoF model is employed as the prediction model in the MPC controller.

Finally, practical constraints are incorporated to ensure safety and feasibility. Input saturations are imposed: U_4 is bounded between the minimum and maximum thrust levels, while $U_{1..3}$ are limited according to actuator capabilities. Furthermore, the attitude angles ϕ, θ, ψ are constrained within specified limits (soft constraints to prevent excessive tilt), and a hard constraint is applied on altitude $x \geq 0$ to ensure the vehicle remains above the surface. These constraints are explicitly enforced within the MPC framework.

III. MPC Problem Formulation

Model Predictive Control (MPC) is an optimal control approach that solves a constrained optimization problem at each time step, employing the system model to predict future behavior. At each control update, MPC simulates how the state will evolve over a fixed prediction horizon (e.g., several seconds into the future) for various possible control input sequences. It then determines the control sequence that minimizes a cost function (also referred to as the performance index) while satisfying all constraints on inputs and states. Typically, the cost function is quadratic, penalizing deviations from a desired trajectory or target state and large control efforts. In a powered descent scenario, for instance, the cost may penalize tracking errors with respect to the landing target (e.g., final position and velocity errors) as well as excessive fuel consumption or attitude deviations. The optimization yields an optimal control sequence; only the first control action is applied to the spacecraft, and at the next time step the process is repeated. This rolling-horizon strategy explains why MPC is a closed-loop feedback controller [3].

A. Key Features of MPC

Unlike conventional controllers, MPC can handle multivariable control objectives and explicit constraints by design. For example, thrust limits, gimbal angle limits, and no-fly zone constraints can be encoded as inequalities in the optimization problem and are guaranteed not to be violated. This represents a major advantage over classical linear controllers such as PID or LQR, which generally do not account for constraints except through ad-hoc saturations. The predictive nature of MPC enables anticipation of future behavior and proactive action (e.g., initiating a gravity turn or deceleration burn sufficiently early to achieve precise landing). Since the controller foresees the trajectory, it can optimize performance criteria smoothly, such as balancing fuel consumption against tracking error—something difficult to achieve with fixed-gain controllers.

B. Cost Function and Formulation

In the present formulation, MPC employs the discretized LTI model described in the previous section as its prediction model. A quadratic cost function is defined to accumulate weighted state errors and control efforts over the horizon as shown in Eq. (6):

$$J = \sum_{k=0}^{N-1} \Big[(\mathbf{x}(k) - \mathbf{x}_{\text{ref}})^T Q (\mathbf{x}(k) - \mathbf{x}_{\text{ref}}) + \mathbf{u}(k)^T R \mathbf{u}(k) \Big] \tag{6}$$

where N is the horizon length (number of prediction steps), Q, R are weighting matrices, and $\mathbf{x}_{\text{ref}}$ denotes the reference final state (for landing, this could correspond to the zero-velocity, zero-error state at touchdown).

The cost penalizes deviations from the target state throughout the descent as well as excessive control usage. By tuning the elements of Q and R, different objectives can be prioritized (e.g., assigning more weight to vertical velocity error ensures a softer landing, whereas weighting horizontal position errors emphasizes lateral accuracy). The flexibility of MPC tuning is powerful: increasing the weight on a state variable forces tighter tracking of that variable, often at the expense of greater control effort or longer maneuver duration. For instance, increasing the weights on horizontal velocity in the cost compels the rocket to reduce lateral drift more quickly, leading to a different thrust and tilt profile and potentially a slower lateral approach, which might be desirable for precise landing [1].

C. Constraints

The MPC optimization at each cycle is subject to constraints that reflect vehicle and mission limits. Input bounds include thrust and control moment limits, while state constraints encompass attitude limitations and ground collision avoidance ($x \geq 0$). By construction, MPC guarantees adherence to these limits in the planned trajectory. This contrasts with, for example, an LQR controller, which produces a single unconstrained gain; LQR might command control inputs beyond actuator capabilities or attitudes beyond safe limits unless complemented by additional logic. MPC inherently prevents such violations by incorporating constraints directly in the optimization [3].

D. Limitations

The primary drawback of MPC lies in the computational demand of solving an optimization problem at each time step. In earlier space applications, this posed a significant barrier, but advances in processors and algorithms are mitigating this limitation. Specialized QP solvers and embedded convex optimization libraries now enable real-time MPC on flight hardware. Another challenge is the dependency of MPC performance on model accuracy and correct constraint specification. Model mismatches or unmodeled dynamics (e.g., neglected aerodynamic forces) may degrade performance or even destabilize the system if unaccounted for. In practice, safety margins are often included in constraints, and models may be updated adaptively as new data become available. Tuning the cost function weights is another practical difficulty, as weights must balance competing objectives (precision landing, fuel efficiency, smoothness of control). Although this tuning is typically more intuitive than tuning multiple PID gains, it still requires iterations or higher-level optimization. Finally, ensuring guaranteed stability of MPC controllers can be challenging; however, by employing terminal costs and constraints or specific MPC formulations with stability guarantees, closed-loop stability can be ensured in theory.

IV. MATLAB & YALMIP Implementation

The implementation of the MPC controller was carried out in MATLAB through the YALMIP toolbox [4]. The procedure comprised model definition, discretization, controller design, and simulation.

A. Model

The vehicle was modeled as a rigid body with the following parameters:

Parameter	Value
I_{xx}	330.472 kg · m^2
I_{yy}	332.721 kg · m^2
I_{zz}	334.931 kg · m^2
m	919.200 kg
Mars gravity g	3.711m/s^2

The system matrices $A \in \mathbb{R}^{12\times 12}$ and $B \in \mathbb{R}^{12\times 4}$ were constructed according to the linearized dynamics, with the state vector

$$\mathbf{x} = \begin{bmatrix} \phi,\ \dot{\phi},\ \theta,\ \dot{\theta},\ \psi,\ \dot{\psi},\ x,\ \dot{x},\ y,\ \dot{y},\ z,\ \dot{z} \end{bmatrix}^T,$$

and the input vector

$$\mathbf{u} = \begin{bmatrix} U_1,\ U_2,\ U_3,\ U_4 \end{bmatrix}^T,$$

where U_1, U_2, U_3 are the control moments about roll, pitch, and yaw, and U_4 is the thrust force.

The continuous-time state-space model, obtained by linearizing around a nominal hover condition defined by $\phi = \theta = \psi = 0$ and $U_4 = mg$, corresponding to a stationary hover, was discretized using zero-order hold with sampling time $T_s = 0.01\,\text{s}$, yielding the discrete system (A_d, B_d). The output matrices were set as $C = I_{12}$ and $D = 0$.

$$\mathbf{x}_{k+1} = A_d\mathbf{x}_k + B_d\mathbf{u}_k$$

B. MPC Setup

The MPC problem was defined with horizon parameters $N = 20$ (prediction horizon) and $N_c = 5$ (control horizon). The state dimension was $n_x = 12$ and the input dimension $n_u = 4$. The initial condition x_0 was chosen as:

Variable	Position	Velocity
ϕ	0 rad	0 rad/s
θ	0.8863 rad	0 rad/s
ψ	-0.49 rad	0 rad/s
x	1500 m	-75 m/s
y	500 m	40 m/s
z	2000 m	100 m/s

while the target state was set to $\mathbf{x}_{\text{ref}} = \mathbf{0}$.

The quadratic cost function expressed by Eq. (6) was designed with weighting matrices $Q \in \mathbb{R}^{12\times 12}$ and $R \in \mathbb{R}^{4\times 4}$. The state weighting matrix was defined as:

$$Q = 100\cdot\text{diag}(5,\ 0.5,\ 10,\ 0.01,\ 5,\ 0.01,\ 10,\ 15,\ 1,\ 1,\ 1.8,\ 1.8)$$

while the input weighting matrix was chosen as:

$$R = 0.001 \cdot \text{diag}(0.001,\ 0.001,\ 0.001,\ 1).$$

An additional slack variable ϵ_S was introduced to soften state constraints, penalized with weight 0.01.

C. Constraints

The MPC optimization included the system dynamics as equality constraints

$$\mathbf{x}_{k+1} = A_d\mathbf{x}_k + B_d(\mathbf{u}_k - \mathbf{u}_{\text{offset}}),$$

where $\mathbf{u}_{\text{offset}} = [0,\, 0,\, 0,\, mg]^T$ compensates for equilibrium thrust.

State constraints enforced limits on attitude and altitude:

$$|\phi|, |\theta|, |\psi| \leq 2\pi + \epsilon_S, \quad x \geq 0.$$

Input constraints imposed saturation bounds:

$$|U_1|, |U_2|, |U_3| \leq 300\ N \cdot m, \quad U_4 \leq 15000\ N.$$

V. Simulation Results

The simulation was performed over $t_{\text{sim}} = 300\,\text{s}$ with step size T_s. At each iteration, the optimizer provided the control sequence $\{U_0, U_1, \ldots, U_{N-1}\}$, of which only the first element was applied. The resulting closed-loop trajectory $\mathbf{x}(t)$ and control history $\mathbf{u}(t)$ were stored. Accelerations were obtained numerically by differentiating the velocity states.

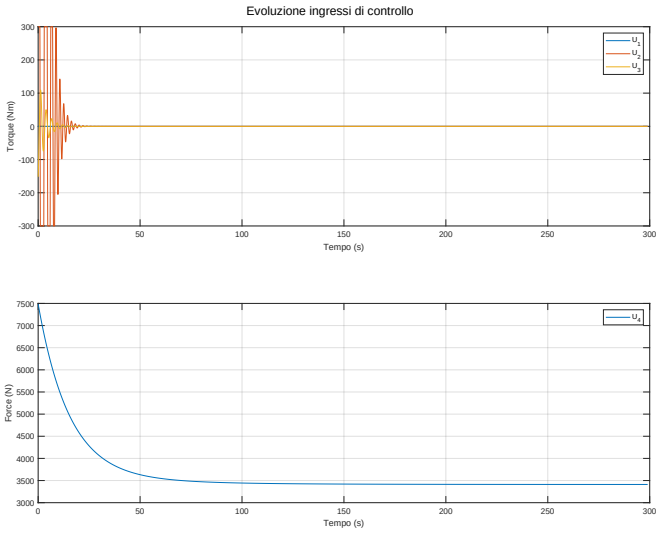

Fig. 2: Control inputs

From the plots, it is evident that the MPC controller successfully guided the vehicle to a soft landing at the origin. The attitude angles remained within limits, and the thrust profile was smooth without excessive oscillations. The velocity profiles show a gradual reduction to zero at touchdown, with accelerations peaking during deceleration burns. Overall, the MPC demonstrated effective trajectory regulation and constraint satisfaction throughout the descent.

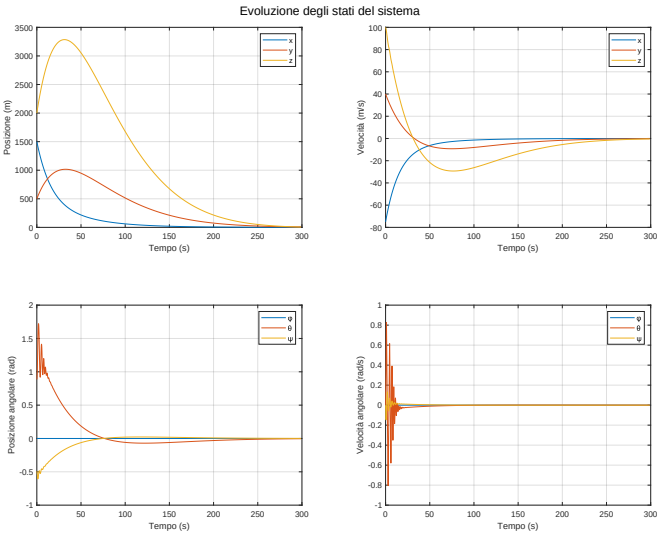

Fig. 3: State evolution

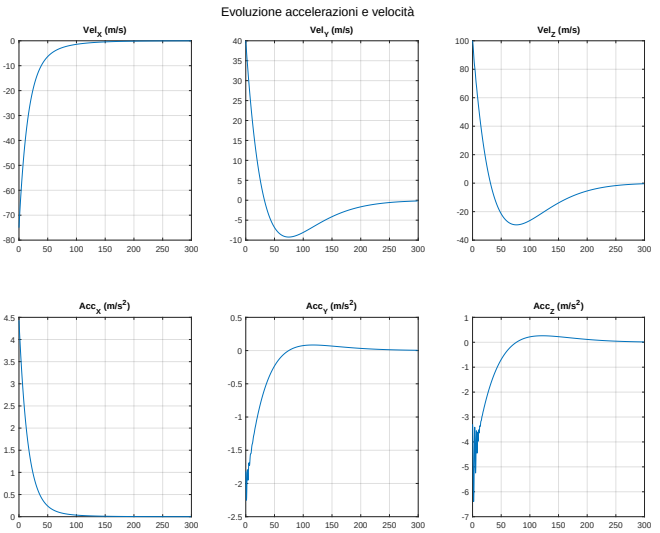

Fig. 4: Velocity and acceleration profiles

References

[1] C. A. Pascucci, S. Bennani, and A. Bemporad, "Model predictive control for powered descent guidance and control," in *2015 European Control Conference (ECC)*, 2015, pp. 1388–1393. [Online]. Available: https://ieeexplore.ieee.org/document/7330732

[2] G. Zaragoza Prous, E. Grustan-Gutierrez, and L. Felicetti, "A decreasing horizon model predictive control for landing reusable launch vehicles," *Aerospace*, vol. 12, no. 2, 2025. [Online]. Available: https://www.mdpi.com/2226-4310/12/2/111

[3] E. N. Hartley, "A tutorial on model predictive control for spacecraft rendezvous," in *2015 European Control Conference (ECC)*, 2015, pp. 1355–1361. [Online]. Available: https://ieeexplore.ieee.org/document/7330727

[4] J. Löfberg, "Yalmip : A toolbox for modeling and optimization in matlab," in *In Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004. [Online]. Available: https://ieeexplore.ieee.org/document/1393890