2015 European Control Conference (ECC)
July 15-17, 2015. Linz, Austria

Model Predictive Control for Powered Descent Guidance and Control

Carlo Alberto Pascucci[1], Samir Bennani[2] and Alberto Bemporad[1]

Abstract— **Future safety critical space missions call for increasing levels of embedded spacecraft autonomy. Spacecraft and mission responsiveness will be highly improved via onboard automation and autonomy functions, simplifying operations and ground control capabilities. Recently developed real-time embedded MPC guidance and control strategies have a great potential for the next generation of high performance reusable ESA launch vehicles and GNC systems. This paper addresses these technologies for real-time embedded MPC covering thrust vectored control that could be proficiently used during the ascent phase as well as on powered descent enabling accurate pin-point landing features.**

I. Introduction

Autonomy, when addressed through the availability of onboard capabilities that resolve in real-time complex mission objectives in the face of unforeseen events and constraints, will largely simplify spacecraft operational modes. Pin-point landing, defined as the ability to land within a hundred meters from a target, is an excellent example about the future challenges in space applications [1], [2]. The science return of planetary missions could be highly increased by improving the precision of autonomous landing. However, despite many efforts done, we are still far reaching an ideal level of precision. A guidance navigation and control (GNC) system capable to withstand the uncertainties encountered during the entry descent and landing (EDL) phases is needed to allow a spacecraft to closely target the most interesting scientific areas. From a control point of view, also the atmospheric ascent phase is strictly related to the pin-point landing. To date, each launch have to be carefully planned with large advance accounting for wide tolerances on all engineering aspects; moreover the launch windows are hardly constrained to the actual weather conditions that can be precisely known only a few hours in advance. These issues have a great impact on the time needed, the resources, and costs involved in launch arrangements. An effective launcher vehicle (LV) control strategy could allow one to reduce the tolerances on the requirements, saving time and avoiding delays often due to weather conditions. Reusability is another key area that could be enabled by a novel GNC architecture. Recovering LV's stages to fly them several times will both reduce costs and increase mission responsiveness [3]. Recently, notable steps forward in this direction have been taken also by private space companies like SpaceX [4] that is currently testing a promising first stage re-entry technology. Focusing on GNC tasks, the use of on-line model-based control strategies with adaptive prediction horizons is a key enabling technology that could lead to unprecedented levels of autonomy for a wide range of space missions. Model predictive control (MPC) is a systematic design approach for controlling multivariable systems, maximizing their performance under various restrictions on input and output variables, that can automatically and smoothly reconfigure to structural or operational changes. The MPC rationale is to use a dynamical model of the process to be controlled to predict its future evolution and choose the best control action. In this respect, MPC can be considered as an on-line adaptive control strategy using prediction models that can be possibly updated at run-time to reflect the actual status of the system such as in case of failures and degradations. In the MPC field, various research groups [5]–[13] as well as previous ESA studies, such as ORCSAT and ROBMPC projects, have already lead to novel control strategies, and software packages, like ESA's MPCTool and MPCSofT, that assessed successful real-time MPC implementations for Martian orbit rendezvous [14] and planetary rover control problems [15], [16]. Continuous advances in more efficient and higher performance computational platforms, combined with the availability of fast and reliable optimization codes, make MPC an even more appealing candidate technology for embedded space applications [17], [18]. Convex optimization solvers are indeed the core of this control framework. In the last few years, advances in the area of the second order cone programming (SOCP) lead to flights demonstrating on-line embedded optimization [19]. This important result relies on the "lossless convexification" technique [20]–[22]. A convex optimization branch that is crucial to MPC is quadratic programing (QP). Newly developed algorithms [23]–[25] and software [26], with dedicated auto-coding functions are enabling the use of MPC in effective on-board prototyping tools that can solve in real time large scale, fast and constrained optimal control problems.

In this paper, starting from the definition of the thrust vectoring control problem, we will go from the theoretical design to the implementation and validation of the proposed MPC controller on an embedded processor, providing simulation examples and execution time measurement to assess the feasibility of the presented approach.

This paper is organized as follows: Section II details the proposed MPC design for thrust vectoring control; in Section III a simplified Mars powered descent case study is presented along with simulation results and consider-

*This work was supported by the European Space Agency through the Network Partnering Initiative programme (NPI), Grant number 4000106153

[1]C. A. Pascucci and A. Bemporad are with IMT Institute for Advanced Studies Lucca, Piazza San Francesco 19, 55100 Lucca, Italy. Email {ca.pascucci,alberto.bemporad}@imtlucca.it

[2]S. Bennani is with ESA-ESTEC, Keplerlaan 1, Noordwijk 2201 AZ, The Netherlands samir.bennani@esa.int

ations about the problem's computational complexity; in Section IV, focusing on performance analysis, we revise our embedded MPC implementation, while in Section V some conclusions are drawn.

II. MPC For Thrust Vectoring Control

From a control point of view, the powered descent (PD) phase for pin-point landing is a thrust vectoring problem. The attitude, the velocity and the position of a LV or a lander can be indeed regulated by means of its thrust vector that is defined as the magnitude and direction of the forces generated by the spacecraft's engines and control surfaces.

A. Prediction Model

In the presented study the vehicle has been modeled as a rigid body with six degrees of freedom (6-DoF). 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 Figure 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:

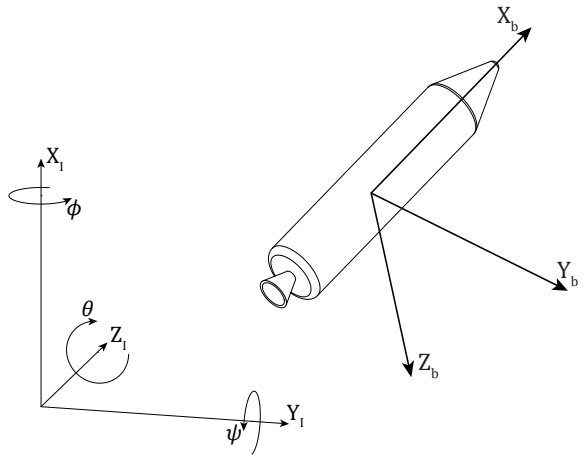

Fig. 1. Inertial $[X_I, Y_I, Z_I]$ and body $[X_b, Y_b, Z_b]$ frames

$$\begin{bmatrix}\ddot{x}\\ \ddot{y}\\ \ddot{z}\end{bmatrix} = \frac{1}{m}\cdot C_I^b \cdot \begin{bmatrix}F_{bx}\\ F_{by}\\ F_{bz}\end{bmatrix} + \begin{bmatrix}g_{Ix}\\ g_{Iy}\\ g_{Iz}\end{bmatrix} \tag{1a}$$

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

where $[\ddot{x}\ \ddot{y}\ \ddot{z}]^\top$ are the body's linear accelerations along the X_I, Y_I, and Z_I inertial frame axes respectively, m is mass, C_I^b is the rotation matrix between the body and the inertial frames, $[F_{bx}, F_{by}, F_{bz}]^\top$ are the forces applied to the body and $[g_{Ix}\ g_{Iy}\ g_{Iz}]^\top$ is the gravity force vector already expressed in the inertial frame. In (1b) $[\ddot{\phi}\ \ddot{\theta}\ \ddot{\psi}]^\top$ are the angular accelerations, $I_{xx}, I_{xy}, I_{xz}, I_{yy}, I_{yz}, I_{zz}$ are the inertia tensor components, $[M_x, M_y, M_z]^\top$ are the moments about the CoG, and $\times$ denotes the cross product operator. While a detailed description of 6-DoF dynamics can be found in [27], considering the application domain, it is safe to assume the following model simplifications: a) no planet rotation; b) flat planet surface; c) uniform gravity field; d) negligible aerodynamic effects; e) diagonal inertia matrix. Moreover regarding the control inputs a special consideration have to be made. Depending on the rocket's design, the spacecraft can have single or multiple nozzles, that can be gimbaled or fixed, and also the thrust magnitude can be predefined or dynamically adjusted. The vehicle can be steered by varying the nozzle's swiveling angle, exploiting the differential thrust or, in a more flexible fashion, by a combination of these methods. Furthermore, during atmospheric flight, fins or other passive aerodynamic control surfaces could be used. The actuators allocation problem can be then addressed by means of a dedicated module or, if needed, it can be included in the optimal control design. Because of the great variability in the rocket's actuation scheme, for the sake of generality in this paper we will consider four virtual control inputs U_1, U_2, U_3, U_4 to pilot the spacecraft, where U_1 is the rolling moment, U_2 is the pitching moment, U_3 is the yawing moment, and U_4 is the thrust force. The main idea behind our control framework is not to design a set of controllers to follow a previously designed optimal trajectory, but rather is to use MPC to optimally steer the vehicle toward a desired state. In other words the trajectory is a controller's outcome and not a flight reference or constraint. Following the EoM (1) and the mentioned simplifying assumptions, by adopting Euler's angles the rocket's dynamical model used the control design can be written as

$$\begin{aligned} &\ddot{x} = \tfrac{U_4}{m}\cos\theta\cos\psi - g, \quad \ddot{y} = \tfrac{U_4}{m}\cos\theta\sin\psi, \quad \ddot{z} = -\tfrac{U_4}{m}\sin\theta \\ &\ddot{\phi} = \tfrac{U_1}{I_{xx}}, \quad \ddot{\theta} = \tfrac{U_2}{I_{yy}}, \quad \ddot{\psi} = \tfrac{U_3}{I_{zz}} \end{aligned} \tag{2}$$

By linearizing (2) with $U_4 = m\cdot g$ and $[\phi, \theta, \psi]^\top = [0,0,0]^\top$, such that the thrust compensates for the weight force and the rocket nose is pointing upwards, it is possible to obtain the linear time invariant (LTI) model that, once discretized with a sampling time of $T_s = 0.01\ s$, will be used as prediction model in the following MPC design. This linearization point is similar to the hovering condition in rotary winged aircraft, that is the rocket stands still at mid air balancing the externally applied constant forces. For this optimal control problem we define a constraint set on both state and input variables. Specifically, hard box constraints on $U_i,\ i = 1,\ldots,4$, soft box constraints on attitude angles θ, ϕ, ψ, plus an additional hard one on the minimum altitude, $z \geq 0$. Depending on the mission scenario this set could be extended to include for instance constraints on angular velocity to prevent excessive mechanical stress on the rocket.

B. Linear MPC Problem Formulation

We can now formulate the MPC problem through equations (3a), and (3b),

$$\begin{cases} \chi(k+1) = A\chi(k) + Bu(k) + f \\ \eta(k) = E_\eta \chi(k) + H_\eta u(k) + P_\eta \Delta u(k) \\ \gamma(k) = E_\gamma \chi(k) + H_\gamma u(k) + P_\gamma \Delta u(k) \end{cases} \tag{3a}$$

$$\begin{cases} \min\limits_{\Delta u, \varepsilon_1, \varepsilon_2} \quad \rho_1 \varepsilon_1^2 + \rho_2 \varepsilon_2^2 + \sum\limits_{k=0}^{N-1} (\eta(k) - r(k))^\top (\eta(k) - r(k)) \\ \text{subject to} \\ \Delta u(k) = 0, \ \forall k = N_u, ..., N \\ \gamma(k) \leq \gamma_{max} + V_\gamma \varepsilon_1, \ k = 0, ..., N_{cy} - 1 \end{cases} \tag{3b}$$

where $\chi = [\phi, \dot{\phi}, \theta, \dot{\theta}, \psi, \dot{\psi}, x, \dot{x}, y, \dot{y}, z, \dot{z}]^\top$ is the state vector of the dynamical system, k is the discrete-time instant, $u = [U_1, U_2, U_3, U_4]$ is the control input vector, $\Delta u(k) = u(k) - u(k-1)$ is the input increment, η is the "performance" vector defining what we want to optimize, γ is the constraint vector, and E, H, P are matrices of opportune dimensions used to shape MPC performance and constraints. In Eq. (3b) ρ_1 and ρ_2 are weighting terms for the slack variables ε_1 and ε_2 respectively, used to deal with infeasibility, N is the prediction horizon, r is the reference vector, N_u is the control horizon, N_{cy} is the constraint horizon, γ_{max} is the constraint upper-bound vector, and V_γ defines whether a constraint is hard ($V_\gamma = 0$) or soft ($V_\gamma > 0$). At each time instant k, the linear MPC controller sets $u(k) = u(k-1) + \Delta u^*(k)$, where $\Delta u^*(k)$ is the first element of the computed optimal input sequence. We complete the MPC setup by mapping equations (3a), and (3b) in the QP problem

$$\begin{cases} \min\limits_{w} \quad \frac{1}{2} w^\top H w + c^\top w \\ \text{subject to} \quad Gw \leq b \end{cases} \tag{4}$$

In (4) w represents the optimization vector, H is the Hessian matrix, c the linear weighting vector, G is the matrix of linear constraints, and b is the right-hand-side vector with dimensions

$$\begin{aligned} &H \in R^{\,nvar \times nvar} \quad c \in R^{\,nvar} \quad G \in R^{\,ncon \times nvar} \\ &b \in R^{\,ncon} \quad nvar = n_u \cdot N_u \quad ncon = N_{cy} \cdot n_c \end{aligned} \tag{5}$$

where n_u is the number of control inputs, and n_c is the number of constraints, corresponding to the rows in the E_γ matrix.

III. Mars Powered Descent Case Study

We test the MPC setup (3) in a simulation scenario similar to the one detailed in [28]. The powered descent is the final step in the EDL sequence that starts after the release of the parachute and terminates with the touchdown. Firing the rocket engine, dynamically adjusting the thrust vector, the controller has to guide the lander to the desired position, computing the optimal descent trajectory and the actuation profile on-line in real-time. The simulation starts with the lander at $[x, y, z]^\top = [1500, 500, 2000]^\top$ m, the initial velocity vector is $[\dot{x}, \dot{y}, \dot{z}]^\top = [-75, 40, 100]^\top$ m/s, and the initial spacecraft's attitude is aligned with the velocity vector having the nozzle(s) pointing downward leading to $[\phi, \theta, \psi]^\top = [0, 0.8863, -0.49]^\top$ rad. Without loss of generality the set-point is zero on all state variables. Regarding the environment and the lander parameters, Mars gravity is set to $g = 3.711$ m/s^2, the spacecraft mass is $m = 919.200$ Kg, while the principal components of the inertia matrix are $I_{xx} = 330.472$ $Kg \cdot m^2$, $I_{yy} = 332.721$ $Kg \cdot m^2$, and $I_{zz} = 334.931$ $Kg \cdot m^2$.

A. Simulation Results

The MPC horizons are set to $N = 20$, $N_u = 1$, $N_{cy} = 5$. By shaping the performance vector $\eta(k)$ in (3a) by means of E_η, H_η, and P_η matrices it is possible to focus the controller action on different aspects. For the simulations shown in Figure 2, the performance variable η in (3a) is defined as

$$\begin{aligned} \eta \quad = \quad & \sqrt{[W_\phi \ W_{\dot{\phi}} \ W_\theta \ W_{\dot{\theta}} \ W_\psi \ W_{\dot{\psi}} \ W_x \ W_{\dot{x}} \ W_y \ W_{\dot{y}} \ W_z \ W_{\dot{z}}]}\chi \\ & + \sqrt{[W_{U1} \ W_{U2} \ W_{U3} \ W_{U4}]}u + \sqrt{[W_{\Delta U1} \ W_{\Delta U2} \ W_{\Delta U3} \ W_{\Delta U4}]}\Delta u \end{aligned} \tag{6}$$

where $W_\phi = W_\theta = W_\psi = 5$; $W_{\dot{\phi}} = 0.5$ $W_{\dot{\theta}} = W_{\dot{\psi}} = 0.01$; $W_x = 10$; $W_y = W_z = 1$; $W_{\dot{x}} = 15$; $W_{\dot{y}} = W_{\dot{z}} = 0.5$; $W_{U1} = W_{U2} = W_{U3} = W_{U4} = 0$; $W_{\Delta U1} = W_{\Delta U2} = W_{\Delta U3} = W_{\Delta U4} = 0$. Varying the weight on a state variable affects the effort and the speed in tracking the corresponding reference cf. Figure 3 in where only $W_{\dot{y}}$ and $W_{\dot{z}}$ were changed by setting them as $W_{\dot{y}} = W_{\dot{z}} = 1$. When increasing the weights on $\dot{x}$ and $\dot{y}$ the spacecraft slows down faster, implying a different thrust modulation, an alternative attitude time evolution, and a slower translation on the x and y axes toward the landing point. Depending on the application scenario, it may be advisable to adjust the weighting factors to better meet mission requirements. A detailed analysis about possible tuning for $\eta(k)$ is beyond the purpose of this paper, we will rather focus on the key parameters in the QP that have the largest impact on the computational effort required to converge to a solution. To this end, some details about the selected QP solver are needed.

B. Computational Complexity

For the implementation of the MPC controller we use the accelerated dual gradient projection (GPAD) algorithm of [23] for solving the QP (4). The method is particularly suitable for embedded applications in that it is extremely simple to program and involves only sums, products and comparisons. Hence, it is a code that is very easy to verify, a fact of major importance for aerospace applications. Additionally, GPAD comes with an *a priori* complexity certification, that is an upper bound that can be computed in advance on the maximum number of iterations required to terminate the algorithm within a desired accuracy. One additional detail is that GPAD assumes a strictly convex primal problem, (i.e., the primal Hessian H in (4) needs to be positive definite). The main drawbacks of dual methods are that a) if stopped prematurely a feasible solution will not be produced, b) the matrix vector operations require

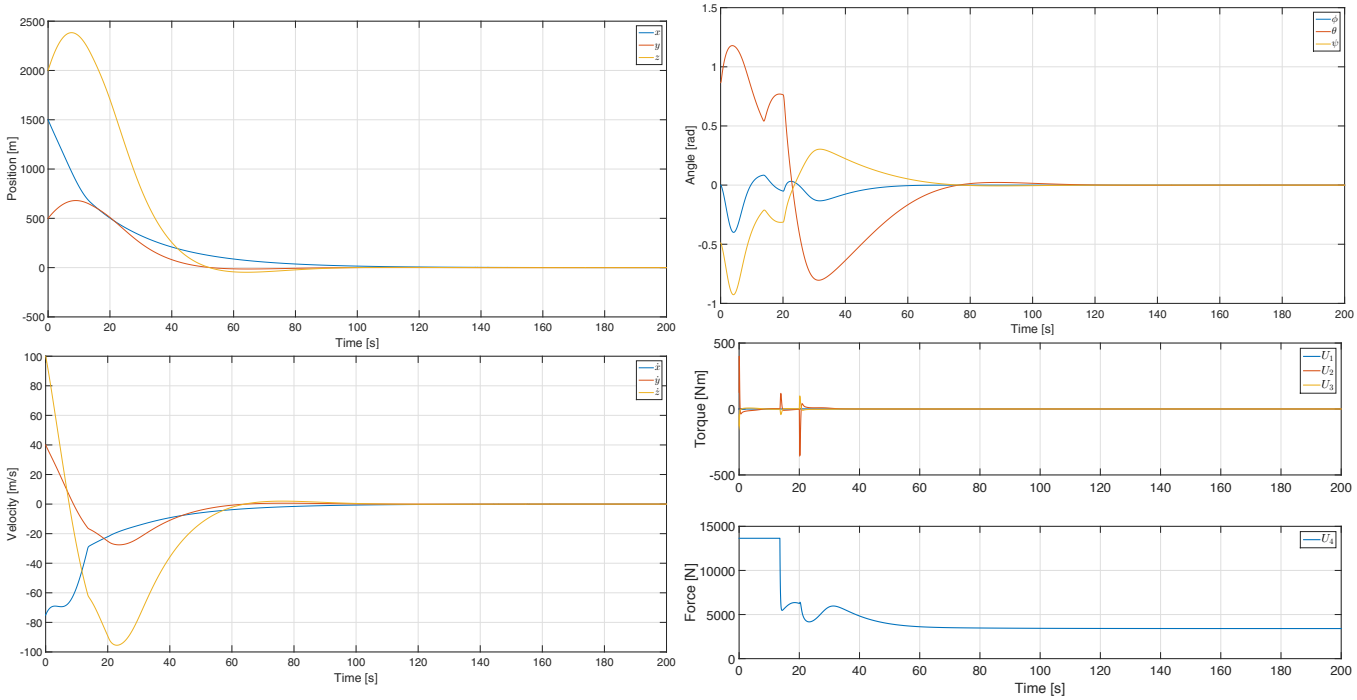

Fig. 2. Baseline simulation results

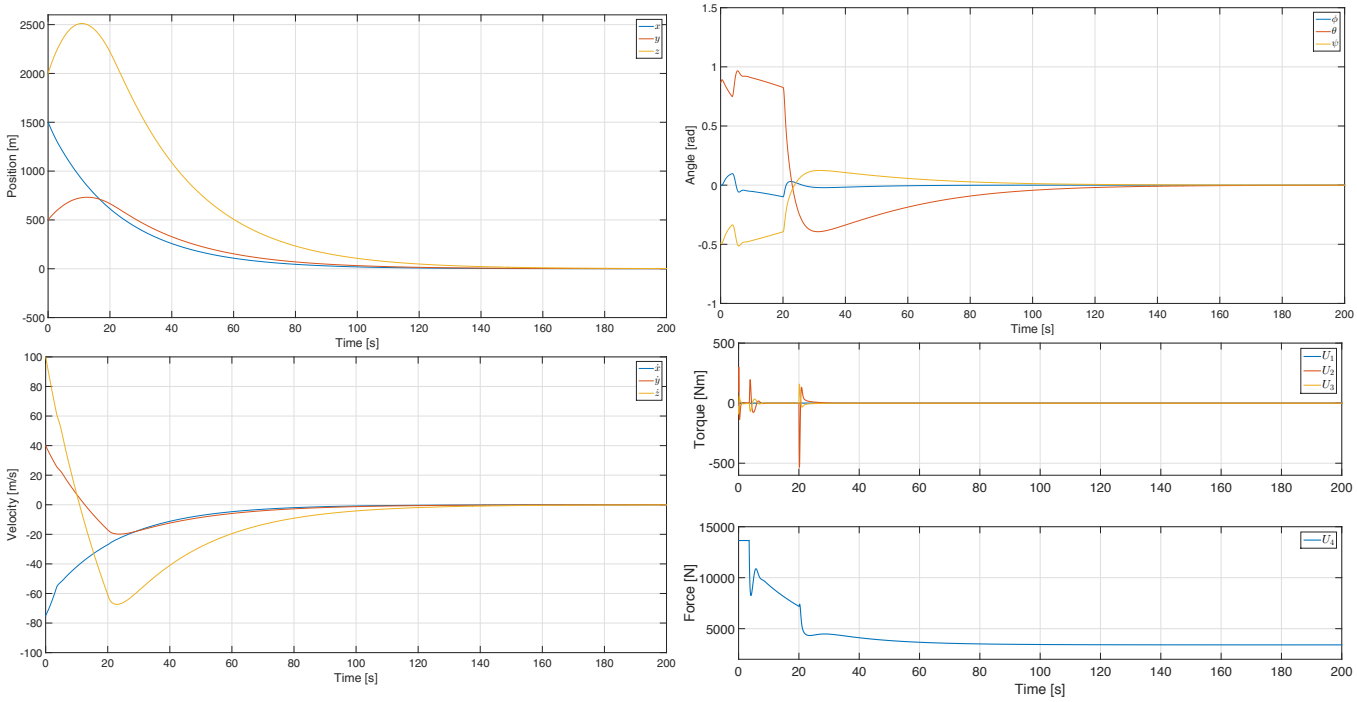

Fig. 3. Alternative weighting simulation results

the computation of the dual Hessian matrix $M = GH^{-1}G^{\top}$ which is generally of higher dimension than the primal one, and c) as for primal methods, the convergence rate is sensitive to problem scaling. To face these issues we have a) to ensure that the computational time for the maximum number of iterations will not exceed the given controller sampling time, b) minimize the QP matrices dimension, and c) precondition the QP problem to enforce numerical stability. It is worth noting that the only parameters that affect uniquely the required number of iterations are the upper bounds on the primal optimality and infeasibility tolerances $\varepsilon_V, \varepsilon_G$. By setting $\varepsilon_V = \varepsilon_G = 10^{-1}$, the QP solver converges within $n_{iter_{max}} = 6$ iterations, while if $\varepsilon_V = \varepsilon_G = 10^{-8}$ the same QP problem requires $n_{iter_{max}} = 90$ iterations in the worst case. Varying such bounds influences the quality of the solution, so a trade off must be found. Recently [29] showed

that MPC controllers can preserve stabilization properties in spite of numerical inaccuracy. In the simulations presented in Section III-A we used the tolerances $\varepsilon_V = 10^{-4}$ and $\varepsilon_G = 10^{-4}$, which lead to $n_{iter_{max}} = 23$ iterations maximum to converge without showing any noticeable degradation in the solution's quality with respect to smaller tolerance values. All the other variables, such as the MPC's horizons and the enforced constraints, affect not only the convergence rate, but also the dimension of the QP problem as detailed in Eq. (5). For the GNC application examined in this paper we have $nvar = 4$ optimization variables, with a total number of constraints $ncon = 35$ or $ncon = 55$ if including also box constraints on roll and pitch on top of the minimum altitude limit and the lower and upper bounds for the control inputs. Concerning problem scaling, let the dual QP problem be defined as

$$\begin{cases} \min\limits_{v} \quad \frac{1}{2} v^\top M v + d^\top v \\ \text{subject to} \quad v \geq 0 \end{cases} \tag{7}$$

where v, is the dual optimization variable, $M = GH^{-1}G^\top$ and $d = GH^{-1}c + b$. A simple and effective way to scale is to define the following matrix [30]

$$P = diag\left(\frac{1}{\sqrt{M_{ii}}}\right) \tag{8}$$

let $v = Pv_s$, and consider the scaled dual QP problem

$$\begin{cases} \min\limits_{v_s} \quad \frac{1}{2} v_s^\top (PMP) v_s + d^\top P v_s \\ \text{subject to} \quad v_s \geq 0 \end{cases} \tag{9}$$

Without scaling the dual QP as in (9), for the same problem the maximum number of iterations grows from $n_{iter_{max}} = 23$ to $n_{iter_{max}} = 1337$.

IV. Embedded MPC

To asses the performance of the proposed MPC design we performed Processor In the Loop (PIL) simulations [31] on a popular embedded board, namely the BeagleBone Black (BBB). In these tests, while Simulink computes the plant's response, the embedded processor is running the MPC controller. The core of the BBB is an ARM Cortex-A8 processor running at 1 GHz. The embedded target support available in Matlab R2014b provides an easy and effective way to generate C code out of the EML Simulink block used to implement the controller. We selected the most computational demanding phase of the PD and measured the CPU time required by the MPC controller running on the board. We tested it against different problem sizes and number of iterations. The results are detailed in Table I. As baseline we compared the embedded CPU results with a MEX version of the controller running on an Intel Xeon E5507 clocked at 2.27 GHz. All the time values are expressed in milliseconds. We recall that the control loop frequency is $100Hz$ hence each single MPC step cannot take longer than $T_{max} = 10ms$.

In Figure 4 we show the measured execution time for the QP problem detailed in the first row of Table I. The time variance among different runs is mainly due to the processor

TABLE I
Elapsed CPU time

ncon	n_{iter}	BBB		Xeon	
		T_{max}	T_{avg}	T_{max}	T_{avg}
35	23	0.66	0.51	0.0153	0.0152
55	83	3.18	2.9	0.086	0.085
140	23	1.86	1.77	0.075	0.07
220	74	8.67	8.31	0.34	0.3

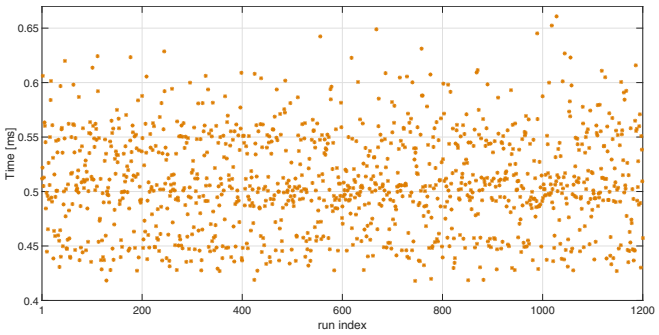

Fig. 4. Timing measurements on the BeagleBone Black board

architecture [32] and to the non-hard real-time nature of tasks created by the PIL verification framework provided by Simulink. Clearly, in a production environment hard real-time performances must be ensured. The longer the job has to be executed, the higher is the probability for the task to be preempted, which can lead to deadline miss and catastrophic consequences on the GNC side. In Figure 5 we show an example of this behavior. While the general trend tracks the number of iterations of the QP solver, the spikes in the recorded task execution time are due to the Linux scheduler that has to distribute the available computational resources among all the concurrently running processes. When the CPU resources are assigned to a different process, the MPC task is suspended and then resumed when the CPU is newly available. This wait time translates in delays in the execution of the MPC code that can lead to a violation of the previously fixed $10ms$ time constraint.

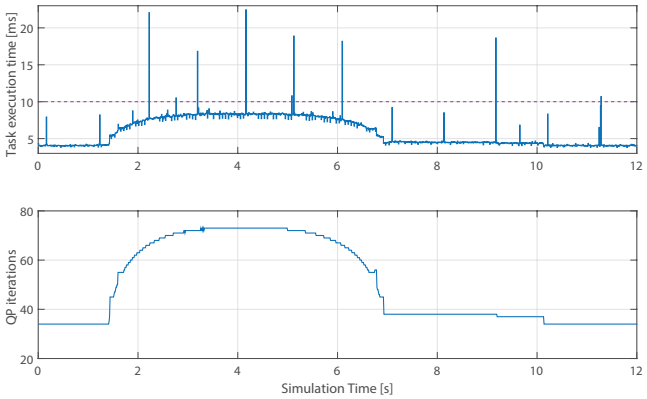

Fig. 5. Spikes due to task preemption

It is worth noting that also the code executing on the BBB is using double-precision floating point arithmetic, switching to single-precision will cut the timing at least by a factor of two. For the $ncon = 35,\ n_{iter} = 23$ test the time is $T_{max} = 0.24ms,\ T_{avg} = 0.14ms$. No attempt was done to improve and speed-up the automatically generated code; significant improvements can be achieved by coding solvers directly in C or by adopting different QP solvers [33].

V. Conclusions

This paper has shown how to take advantage of MPC techniques not only for path following but also to guide and control a lander to a desired position without a pre-configured path, taking into account the full system dynamics and actuator and state constraints. Recently developed QP solvers, such as GPAD, and embedded processors like the ARM Cortex-A family are key enabling technologies to meet the real-time requirements for spacecraft control applications, releasing linear MPC from being bound to slow processes and high performance computing platforms. The LTI nature of the models used in this paper did not consider mass budget and the achieved landing ellipse; the investigation has been initiated in order to understand and benchmark the numerical performance of MPC on various targets. Ongoing and future work will include actuation delays, wind and other external disturbances along with a full linear parameter varying (LPV) setup. As this involves the extra load of constructing the QP problem on-line, a primary concern will be guaranteeing it can be still implemented in low-power embedded platforms.

References

[1] A. A. Wolf, J. Tooley, S. Ploen, M. Ivanov, B. Acikmese, and K. Gromov, "Performance trades for Mars pinpoint landing," in *IEEE Aerospace Conference*, 2006.

[2] B. A. Steinfeldt, M. J. Grant, D. M. Matz, R. D. Braun, and G. H. Barton, "Guidance, navigation, and control technology system trades for Mars pinpoint landing," *Matrix*, vol. 2, p. 1, 2008.

[3] F. Falempin, "The fully reusable launcher: a new concept asking new visions," in *12TH AIAA International Space Planes And Hypersonic Systems And Technologies*. AIAA, 2003.

[4] E. Musk. (2014) SpaceX. [Online]. Available: http://www.spacex.com

[5] A. Bemporad, C. A. Pascucci, and C. Rocchi, "Hierarchical and hybrid model predictive control of quadcopter air vehicles," in *Analysis and Design of Hybrid Systems*, vol. 3, no. 1, 2009, pp. 14–19.

[6] A. Bemporad and C. Rocchi, "Decentralized hybrid model predictive control of a formation of unmanned aerial vehicles," in *Proc. 18th IFAC World Congress, Milano, Italy*, 2011, pp. 11 900–11 906.

[7] C. A. Pascucci, A. Bemporad, S. Bennani, and M. Rotunno, "Embedded MPC for space applications," in *2nd IAA Conference on Dynamics and Control of Space Systems*, 2014.

[8] W. B. Dunbar, M. B. Milam, R. Franz, and R. M. Murray, "Model predictive control of a thrust-vectored flight control experiment," in *Proc. 15th IFAC World Congress on Automatic Control*, 2002.

[9] R. Arthur and H. Jonathan, "Performance evaluation of rendezvous using model predictive control," in *Proc. AIAA Guidance, Navigation, and Control Conference*, 2003.

[10] ——, "Decentralized model predictive control of cooperating UAVs," in *Proc. 43rd IEEE Conference on Decision and Control*, vol. 4, 2004, pp. 4286–4291.

[11] E. Hartley, "Model predictive control for spacecraft rendezvous," Ph.D. dissertation, University of Cambridge, UK, 2010.

[12] M. Wood and W.-H. Chen, "Regulation of magnetically actuated satellites using model predictive control with disturbance modelling," in *Proc. IEEE Int. Conf. on Networking, Sensing and Control*, 2008, pp. 692–697.

[13] Ø. Hegrenæs, J. T. Gravdahl, and P. Tøndel, "Spacecraft attitude control using explicit model predictive control," *Automatica*, vol. 41, no. 12, pp. 2107–2114, 2005.

[14] M. Saponara, V. Barrena, A. Bemporad, E. Hartley, J. Maciejowski, A. Richards, A. Tramutola, and P. Trodden, "Model predictive control application to spacecraft rendezvous in Mars sample return scenario," in *Progress in Flight Dynamics, Guidance, Navigation, Control, Fault Detection, and Avionics*, vol. 6. EDP Sciences, 2013, pp. 137–158.

[15] G. Binet, R. Krenn, and A. Bemporad, "Model predictive control applications for planetary rovers," in *Proc. 11th International Symposium on Artificial Intelligence and Robotics in Space (iSAIRAS)*, 2012, pp. 4–6.

[16] R. Krenn, A. Gibbesch, G. Binet, and A. Bemporad, "Model predictive traction and steering control of planetary rovers," in *Proc. 12th Symposium on Advanced Space Technologies in Automation and Robotics (ASTRA)*, 2013.

[17] A. Guiggiani, I. V. Kolmanovsky, P. Patrinos, and A. Bemporad, "Constrained model predictive control of spacecraft attitude with reaction wheels desaturation," in *Proc. European Control Conference*, Linz, Austria, 2015.

[18] ——, "Fixed-point constrained model predictive control of spacecraft attitude," in *Proc. American Control Conference*, Chicago, IL, 2015, http://arxiv.org/abs/1411.0479.

[19] D. P. Scharf, M. W. Regehr, G. M. Vaughan, J. Benito, H. Ansari, M. Aung, A. Johnson, J. Casoliva, S. Mohan, D. Dueri, *et al.*, "ADAPT demonstrations of onboard large-divert guidance with a VTVL rocket," in *IEEE Aerospace Conference*, 2014, pp. 1–18.

[20] B. Acikmese and S. R. Ploen, "Convex programming approach to powered descent guidance for Mars landing," *Journal of Guidance, Control, and Dynamics*, vol. 30, no. 5, pp. 1353–1366, 2007.

[21] B. Açıkmeşe and L. Blackmore, "Lossless convexification of a class of optimal control problems with non-convex control constraints," *Automatica*, vol. 47, no. 2, pp. 341–347, 2011.

[22] D. Dueri, J. Zhang, and B. Açikmese, "Automated custom code generation for embedded, real-time second order cone programming," in *Proc. IFAC World Congress*, 2014.

[23] P. Patrinos and A. Bemporad, "An accelerated dual gradient-projection algorithm for embedded linear model predictive control," *IEEE Trans. Automatic Control*, vol. 59, no. 1, pp. 18–33, 2014.

[24] B. O'Donoghue, G. Stathopoulos, and S. P. Boyd, "A splitting method for optimal control." *IEEE Trans. Control Systems Technologies*, vol. 21, no. 6, pp. 2432–2442, 2013.

[25] P. Patrinos, L. Stella, and A. Bemporad, "Douglas-Rachford splitting: Complexity estimates and accelerated variants," in *Proc. 53rd Conf. on Decision and Control*, Los Angeles, CA, 2014, pp. 4234–4239.

[26] J. Mattingley and S. Boyd, "CVXGEN: a code generator for embedded convex optimization," *Optimization and Engineering*, vol. 13, no. 1, pp. 1–27, 2012.

[27] A. L. Greensite, *Analysis and design of space vehicle flight control systems*. New York: Spartan Books, 1970, vol. 11.

[28] L. Blackmore, B. Açıkmeşe, and D. P. Scharf, "Minimum-landing-error powered-descent guidance for Mars landing using convex optimization," *Journal of guidance, control, and dynamics*, vol. 33, no. 4, pp. 1161–1171, 2010.

[29] M. Rubagotti, P. Patrinos, and A. Bemporad, "Stabilizing linear model predictive control under inexact numerical optimization," *IEEE Trans. Automatic Control*, vol. 59, no. 6, pp. 1660–1666, 2014.

[30] D. P. Bertsekas, *Convex optimization theory*. Athena Scientific Belmont, MA, 2009.

[31] Mathworks. (2014) Software and processor-in-the-loop (SIL and PIL) simulation. [Online]. Available: http://nl.mathworks.com/help/rtw/examples/software-and-processor-in-the-loop-sil-and-pil-simulation.html

[32] M. Watkins and C. Betancourt. (2014) Ensuring real-time predictability. [Online]. Available: http://www.ti.com/lit/wp/spry264/spry264.pdf

[33] G. Cimini, D. Bernardini, A. Bemporad, and S. Levijoki, "Online model predictive torque control for permanent magnet synchronous motors," in *IEEE Int. Conf. on Industrial Technology*, Seville, Spain, 2015.