SpaceOps-2021,13,x1517

Processor-In-the-Loop Validation of AI-aided Algorithms for On-Board Autonomous Operations

Stefano Silvestrini[a]*, Michèle Lavagna[a]

[a] Department of Aerospace Science and Technology, Politecnico di Milano, Via La Masa, Milano, Italy, stefano.silvestrini@polimi.it, michelle.lavagna@polimi.it
* Corresponding Author

Abstract

The engineering work is definitely not finalized if the algorithms are not tested in relevant hardware. Relevant hardware means some kind of piece of actual technology that could in principle be implemented in satellite missions, according to the size, the mass and the computational power. In order to increase the Technology Readiness Level of neural-based algorithms for distributed GNC developed at Politecnico di Milano - ASTRA research group, the work presents the Processor-In-the-Loop testing campaign executed with relevant hardware: a micro-controller unit and a single-board computer with similar computational power with respect to flight-hardware. An end-to-end autocoding procedure has been developed to transition from Model-In-the-Loop simulations to Processor-In-the-Loop validation. The tests were deemed successful by evaluating the execution times, resource utilization and achieved accuracy.

Keywords: PIL, coding, on-board, validation, GNC

Acronyms/Abbreviations

ANN Artificial Neural Networks
GNC Guidance Navigation and Control
MCU Micro Controller Unit
PIL Processor-In-the-Loop

1. Introduction

Artificial Intelligence represents the new frontier for autonomous robotics algorithms, thanks to the outstanding capability of adapting and generalizing responses to unforeseen events. Moreover, the Artificial Intelligence paradigm can be employed to alleviate the workload of ground-based control, making the planning and scheduling algorithms smarter. In the space domain, the implementation of AI-based is still in the early phases. This is due to the fact that AI-based algorithm are seldom considered as unpredictable, especially when working outside the training domain. In addition, limited computational resources available on-board require sophisticated design of the algorithm to prevent heavy computational loads without losing the benefit of AI. This paper presents a set of neural-aided algorithms employed for on-board spacecraft GNC, which present hybrid architectures between classical approaches and novel AI-based techniques developed at Politecnico di Milano - ASTRA research group [1][2][4][5]. The goal of such algorithms is to support the traditional GNC algorithms, solving the identified shortcomings using neural-based structures. It has been demonstrated that hybrid architectures often present superior performance with respect to both traditional approaches and full AI-based algorithm. The AI-based algorithms are developed for Formation Flying applications in order to enhance the capabilities and the on-board autonomy of the spacecraft. The peculiar environment of proximity operations entail risky operations, characterized by small relative distances and fast dynamics. The ground-based operation paradigm inserts delays in the execution and planning of relative maneuvers that can lead to catastrophic events. For instance, in a distributed architecture the agents have limited knowledge of the neighboring agents. Thus, the spacecraft plan their own trajectories based on relative measurements. The guarantee for collision-free reconfigurations is hardly achievable on-board through classic methods. The human intervention is critical to carefully avoid colliding trajectories to take place. Artificial Intelligence helps in granting superior autonomy performance in order to carry out the tasks that would be otherwise performed by ground control (i.e. collision-free trajectory planning). A GNC planning algorithm for autonomous operations is developed. The Navigation algorithm uses an Extended Kalman Filter, where the prediction step is performed using a neural-reconstructed dynamics. This dynamical model comprises two terms: an analytical expression leveraging on physical understanding of the equations of motions and an additional term learnt and estimated by a Radial-Basis-Function

Neural Network. The latter term entails all the unmodelled perturbations, which can be completely unknown before the spacecraft departure. The Guidance and Control is worked out using a neural-Model Predictive Control which utilizes the neural reconstructed dynamical model to predict and optimize the receding horizon on-board. As previously mentioned, it is critical to proceed with the hardware implementation of such algorithms to test the feasibility of on-board execution. The GNC validation requires the deployment of the algorithms into relevant hardware, which is able to emulate the computational power available onboard. The focus is to validate and verify the feasibility of the developed algorithms implementation into relevant processors and compare their performance with CPU execution. The PIL simulations feature the execution of the GNC algorithms in both Desktop computers and stand-alone boards. The control action is calculated and fed to an orbital dynamics simulator to integrate the motion equations. The communication is performed using USB-serial link. Such process increases the Technology-Readiness-Level of the algorithms, raising it to 3/4. During experimental activities, it has been critical to develop a fast and manageable deployment routine to standardize the process. Once the set-up and porting procedure was fully functional, the rapid prototyping and verification stage could be performed. The paper presents an effective procedure for fast algorithm prototyping. The AI-aided GNC system is deployed to an MCU processor, whose specifications are comparable to flight hardware. The Hardware equipment used in this paper is a microcontroller unit MCU TI LAUNCHXL-F28379D Dual-core architecture. It features:

- Two TMS320C28x 32-bit CPUs
- 200 MHz processor
- IEEE 754 single-precision Floating-Point Unit (FPU)

The CPU frequency can be regarded as representative of the lowest performing on-board computers, which can be easily integrated in nanosatellites. Thus, the preliminary PIL verification and validation of the algorithms was performed using such resource. It is important to remark that only single-precision is supported by the MCU. The Processor-in-the-Loop test by simulating the orbital environment on a dedicated PC, leaving the GNC execution to the MCU. Execution times and numerical accuracy are assessed and compared with on-board requirements, paving the way to extensive testing campaign to increase the TRL of the presented architectures. Preliminary results show that, as expected, the autonomous planner requires more computational time to execute with respect to traditional algorithms. Nevertheless, the resource utilization grows up to $\sim 0.5\%$. As mentioned, the development of the presented algorithms has multiple goals:

- to enhance the on-board autonomy of Formation Flying spacecraft, enabling the system to perform risky operations that can significantly improve the mission outcome;
- to reduce the burden of ground-control, reducing the critical operations executed on-ground.
- being computationally light, online ANN-aided algorithms can be deployed in micro-satellites, where the computational power is limited. This paves the way to new mission concepts employing distributed systems composed of several micro-satellites to enhance mission capabilities, flexibility and reduce space access cost.

2. Processor-In-the-Loop Simulation Setup

The GNC validation requires the deployment of the algorithms into relevant hardware, which is able to emulate the computational power available on-board. The focus is to validate and verify the feasibility of the developed algorithms implementation into relevant processors and compare their performance with CPU execution, as sketched in Fig. 2. The PIL simulations features the execution of the GNC algorithms in both Desktop computers and stand-alone boards. As shown in Fig. 2 the control action is calculated and fed to an orbital dynamics simulator to integrate the motion equations. The communication is performed using USB-serial link. Such process increases the Technology Readiness Level (TRL) level of the algorithms, raising it to 3/4. During experimental activities, it has been critical to develop a fast and manageable deployment routine to standardize the process. This required significant work during the set-up. Once the set-up and porting procedure was fully functional, the rapid prototyping and verification stage could be performed. The boards are shown in Fig. 1. The boards have been selected according to performance standard that are available both for nanosatellite missions as well as larger spacecraft.

Figure 1: Boards used for PIL validation

2.1 Microcontroller Unit

The Hardware equipment used in this paper is a microcontroller unit MCU TI LAUNCHXL-F28379D Dual-core architecture. It features:

- Two TMS320C28x 32-bit CPUs
- 200 MHz processor
- IEEE 754 single-precision Floating-Point Unit (FPU)

The CPU frequency can be regarded as representative of the lowest performing on-board computers, which can be easily integrated in nanosatellites. Thus, the preliminary PIL verification and validation of the algorithms was performed using such resource. It is important to remark that only single-precision is supported by the MCU. The microcontroller has been selected considering as the main driver both the availability of number of output peripherals and compuational power with respect to similar devices of the same class present on the market. The hardware is developed by Texas Instruments (TI), in particular the model is the F28379D LaunchPad Development Kit from C2000 Real-Time Control MCUs family; electronic specifications and wiring can be found directly in the technical datasheets. The board can be accessed and programmed directly from a personal computer by using the serial connection of a USB cable and the dedicated interfacing software Code Composer Studio, but another great advantage of using this TI board is that a complete support package in present also in the Matlab & Simulink suite, asset that makes possible to program it using a Simulink model with the dedicated Embedded Coder, which generates the necessary code directly from the model. This approach is indeed used in the present work.

The TI LaunchPad can also be programmed directly from Simulink, in fact this approach has been used in the present paper work. The rationale is to generate the C, C ++ code from a Simulink model and to deploy it to the target hardware. The first prerequisite is to work both with the TI CCS IDE and controlSUITE. The following Matlab libraries are then needed and utilized:

1. *MATLAB Coder:* compiler that allows generating C and C++ code from MATLAB code.
2. *Simulink Coder*: compiler that allows generating C and C++ code from Simulink code.
3. *Embedded Coder:* adds support for custom targets to the Simulink Coder.

4. *Embedded Coder Support Package for Texas Instruments C2000 Processors*: adds support for the specific family of microcontrollers, including the F28379D Launch-Pad.

2.2 Single Board Computer unit

The Microcontroller Unit (MCU) works with single-precision floating-point unit. This is a limitation for certain applications where complex and accurate calculations require the double precision. To solve such shortcoming the Hardware suite is equipped also with an open-source single board computer, namely the BeagleBone Black. The BeagleBone Black boots Debian operative system. It is important to remark that at the current time the Support Package of Matlab/Simulink ® supports only Debian 7.9, which is actually a deprecated version of Debian. Nevertheless, BeagleBone Black can boot from MicroSD, where Debian 7.9 was flashed to perform PIL tests. The board features:

- AM335x 1GHz ARM ® Cortex-A8
- 512MB DDR3 RAM
- 4GB 8-bit eMMC on-board flash storage
- 3D graphics accelerator
- NEON floating-point accelerator
- 2x PRU 32-bit microcontrollers

Beside the already mentioned Support Packages for autocoding, the necessary Matlab add-ons are necessary to run PIL simulations with BeagleBone Black Hardware:

1. Embedded Coder Support Package for BeagleBone Black Hardware: enables to create and run Simulink models on BeagleBone Black hardware. The support package includes a library of Simulink blocks for configuring and accessing BeagleBone Black peripherals and communication interfaces.
2. ARM Cortex-A Support from Embedded Coder: support for ARM Cortex-A processors by generating processor-optimized code using Ne10 project libraries (Ne10) with the GNU Compiler Collection (GCC) compiler.

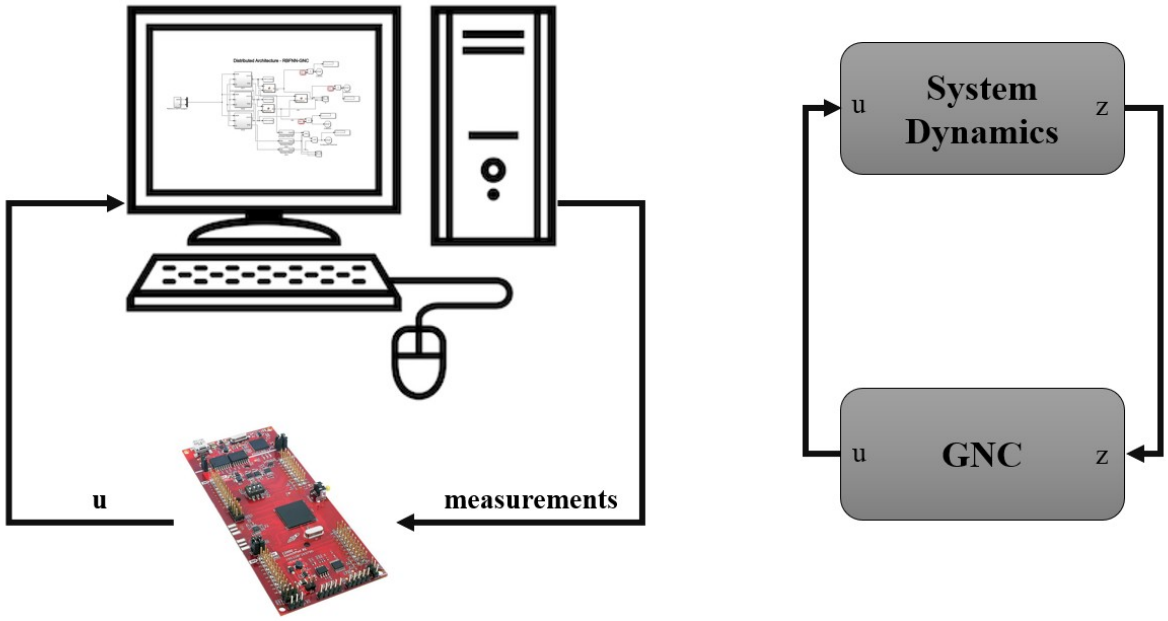

Figure 2: Schematics of Processor-In-the-Loop verification and validation process.

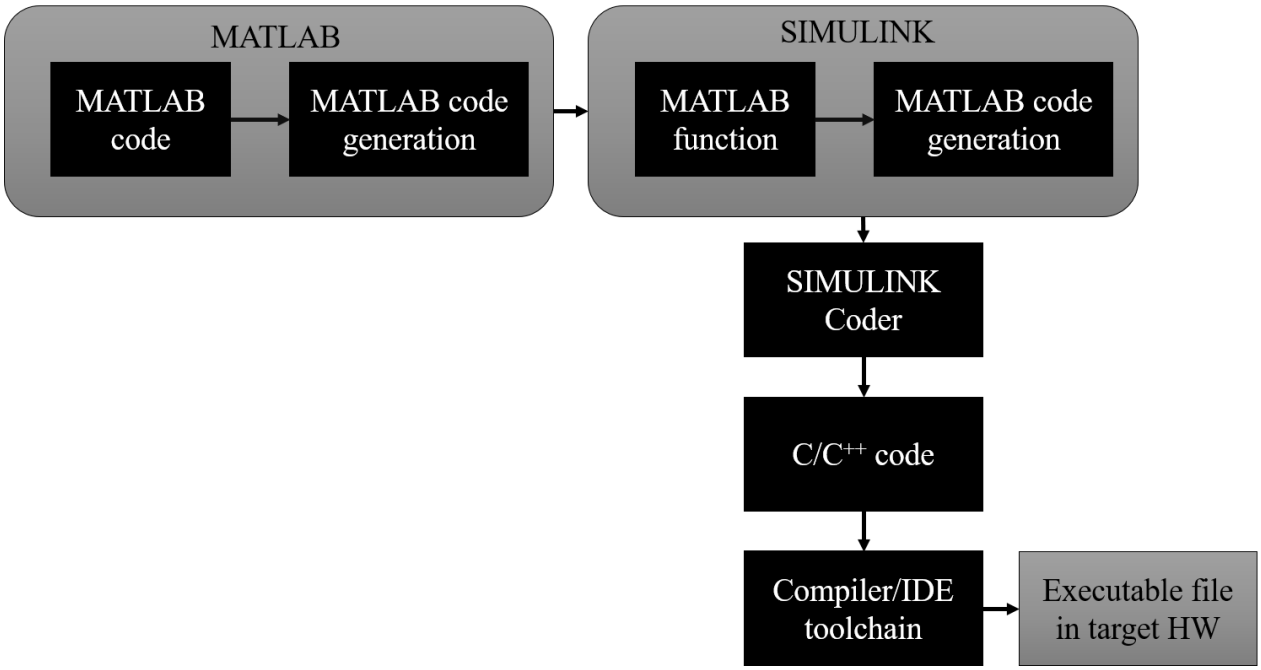

Figure 3: Embedded coder workflow

2.3 Porting Procedure

The porting procedure is executed using the Matlab/Simulink Support Package for both the boards. In particular the Embedded Coder allows the user to autocompile the Simulink code into the target hardware, without requiring deep coding knowledge. This helps the Space community in the rapid prototyping activities. The workflow of the Embedded Coder is schematized in Fig. 3. Running a model in external mode or PIL validation allows to rapidly deploy the code into the hardware and to perform parameter tuning while running. To prepare the deployment of the code, the following settings must be changed:

- Simulation Mode: External or Normal (PIL block generated)
- Model Configuration Parameters → Hardware Implementation → Hardware board
- Target Hardware Resources → Build Options → Device Name
- Target Hardware Resources → Clocking → Tick Use Internal Oscillator
- Target Hardware Resources → SCIA → Pin Ass. (Tx) = GPIO42, Pin Ass. (Rx) = GPIO43 required only for the Microcontroller Unit.
- Target Hardware Resources → External Mode → Set the correct COM port
- For PIL simulations: Code generation → Verification → Advanced parameters → Create block: PIL

The same procedure can be executed to set up the BeagleBone Black target interface. Note that the Support Package from Simulink requires to boot from Debian 7.9 and not later versions. The BeagleBone Black runs Debian Linux distribution, hence it is slightly more cumbersome to set-up all the necessary libraries. First of all, it is important to establish SSH communication with the board. Also, it is critical to connect the board to Internet and run the package update routine (apt-get update). Given that Debian 7.9 is obsolete, it is recommended to force installation of the packages required by the Support Package. After setting up the BeagleBone Black, the Matlab/Simulink target hardware configuration can be run following the aforementioned procedure. The result is a Simulink project where PIL autocoded module is compared against the CPU-executed block, as shown in Fig. 4.

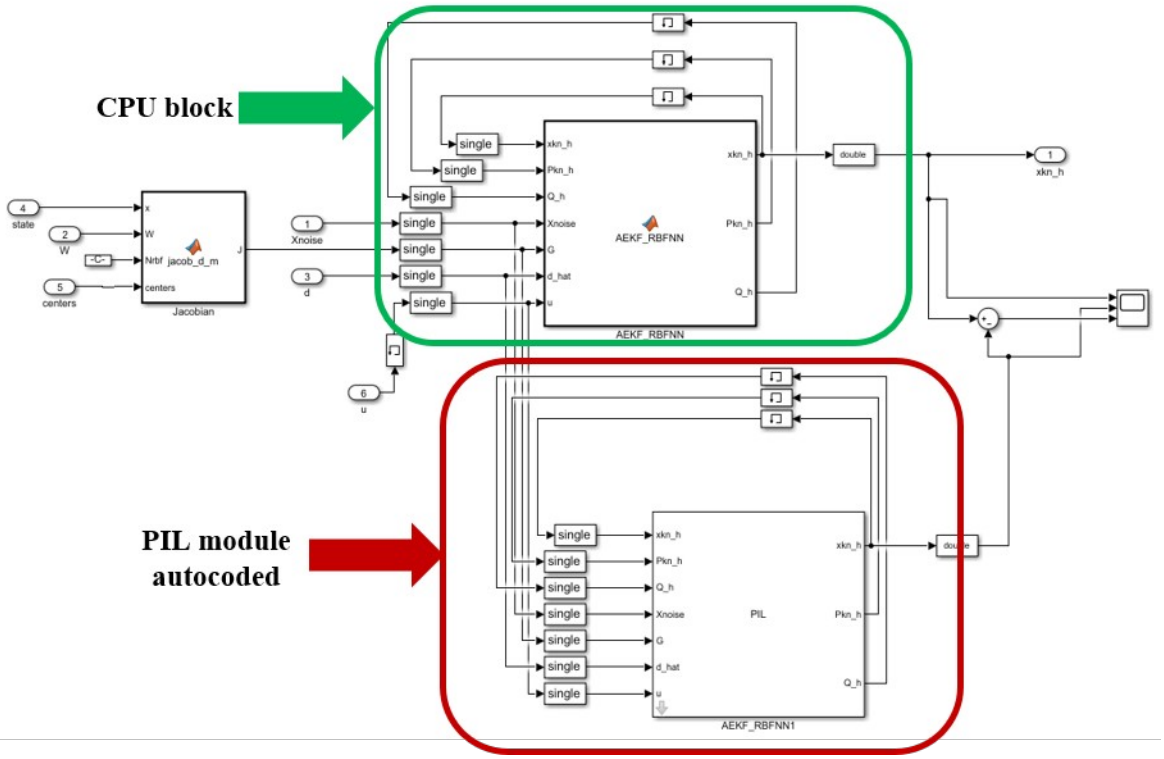

Figure 4: Processor-In-the-Loop (PIL) validation framework in MATLAB/Simulink.

3. Processor-In-the-Loop Validation

The experimental tests with the Processor-In-the-Loop focused on evaluating the feasibility of running the developed algorithms into representative hardware [6]. In particular, the metrics used to evaluate the simulations are defined as:

- Execution times of each sub-routine that build up the algorithms (t e [s])
- Duty cycle and resource utilization with respect to task execution time. It is calculated as the execution times divided by the task allocated time (DC e [%])
- Numerical discrepancy with respect to CPU output (eps). The numerical discrepancy is computed as the difference between the control vector computed in the relevant processor and desktop computer.

The required threshold to be satisfied depend on the mission application and design. Nevertheless, the three metrics are evaluated comprehensively. For instance, the execution times are related with the duty cycle: a threshold of 10% for task duty cycle can be regarded as acceptable. Regarding the numerical discrepancy, it is important to bear in mind that the control action is calculated and fed to actuators in real mission. The sensitivity of such actuators establish the acceptable discrepancy threshold. For this reason, a threshold of 10^{-6} ms^{-2} is set, assuming a conservative value for control sensitivity of current actuators. All the algorithms presented in the paper have been part of the PIL validation campaign. Nevertheless, Recurrent Neural Network still lack support for hardware implementation. This problem is quite extended for representative flight hardware. Indeed, for instance, Intel Openvino framework, used for optimizing code deployment for Intel board (e.g. Myriad 2), does not offer a full support for Recurrent Neural Networks.

3.1 Neural-Network Artificial Potential Field validation

The presented simulations based on the NNAPF algorithm, comprising the whole RBFNN-GNC architecture, were simulated using a Desktop computer using Intel ® Core T M i5-3470 CPU @3.20 GHz. The computational time for a single step execution of the NNAPF is < 50 ms. The NNAPF has been developed for an on-board

applications, hence it is critical to evaluate its computational burden on relevant processor and hardware. The building blocks of such algorithm encompass the online learning, the artificial potential field guidance, the adaptive EKF for navigation and the Lyapunov controller. The average execution times of the autonomous GNC routines are limited and fully compatible with on-board implementation, as reported in Tab. 1. The NNAPF runs at 10 Hz: almost continuous control. For such sample time, the resource utilization is roughly $\sim 2\%$. Fig. 5 shows the discrepancy in the calculated control by the MCU and the Desktop simulator. The values are in the order of $\sim 10^{-12}$ ms^{-2} , which can be assumed to be numerical error, thus validating the PIL test.

Table 1: Average and maximum execution time of GNC routines using a single core TMS320C28x 32-Bit CPUs @200 MHz of TI C2000-Delfino MCUs F28379D

Routine	Average [ms]	Max [ms]
rbfnn()	0.89	0.90
aekf()	0.81	0.82
apf()	0.32	0.32
ctrl()	0.28	0.33

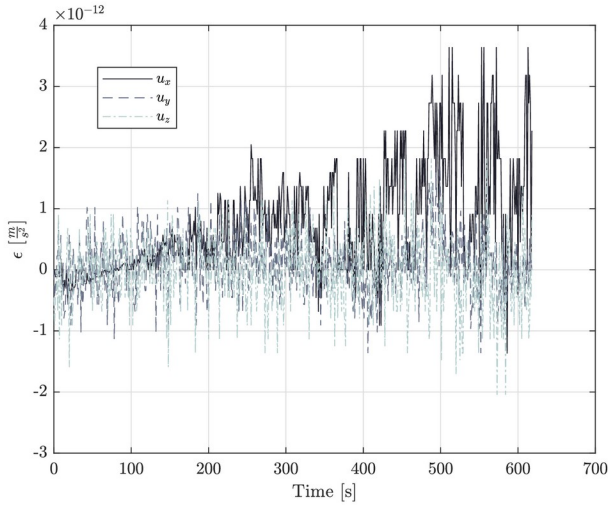

Figure 5: Discrepancy in calculation between PIL and MIL simulations for NNAPF run in MCUs F28379D.

3.2 Model-Based Reinforcement Learning validation

The Model-Based Reinforcement Learning algorithm relies on optimization techniques, in the same fashion as Model Predictive Control. Analogously, the Inverse reinforcement Learning features nested optimization to generate the collision avoidance constraint. On-board optimization is a delicate task to be executed on-board and certainly requires more computational power with respect to NNAPF. To be able to deploy the code to the target hardware with acceptable outcome, double-precision floating-point unit was required. Differently from [3], the MCU does not support double-precision, hence it was necessary to complement the hardware suite with the single-board computer BeagleBone Black. The MBRL entails the artificial neural network learning and the optimization routine for planning and control. In particular, two solvers were used from Matlab suite, namely quadprog and fmincon. The reason is that, if no collision avoidance is implemented, the problem can be recast into quadratic programming formulation that guarantees rapid convergence and existence of the solution. For quadprog the active-set has been chosen as solver algorithm in MATLAB. The collision avoidance constraint is nonlinear, even if convexified. The constraint is nonlinear, hence the problem transforms into a quadratic optimization with nonlinear constraints. The problem is solved using fmincon in MATLAB library implementing sequential quadratic programming. The sqp has been selected for the medium-scale problem in the receding horizon optimization as well as for its efficiency and

robustness. A comparison between the execution of the two methods is reported in Fig. 8. As expected, the nonlinear optimization requires more computational time to execute. The maximum number of iterations was set to 1000 in order not to enter infinite loops in the target hardware. The resource utilization grows up to $\sim 0.5\%$. The comparison of the execution times normalized frequency is shown in Fig. 9. An important remark is that both the distributions are relatively narrow, meaning that the confidence on the mean is high (see Fig. 8) and the outliers still deliver acceptable results. However, one can clearly see from Fig. 9 how the collision avoidance constraint may increase the computational time when it is constraining severely the solution (e.g. very close agents). Indeed, the distribution for fmincon-sqp is more spread out.

The MBRL and IRL have been developed for an on-board applications, hence it is critical to evaluate its computational burden on relevant processor and hardware. Fig. 6 shows the control action calculated by the Single Board Computer (SBC) and delivered to the orbital simulator running in the Desktop computer. The average execution times of the autonomous GNC routines are limited and fully compatible with on-board implementation, as reported in Tab. 2. The MBRL runs at 1/60 Hz, which means that the control is output every minute. For such sample time, the resource utilization is roughly $\sim 0.25\%$.

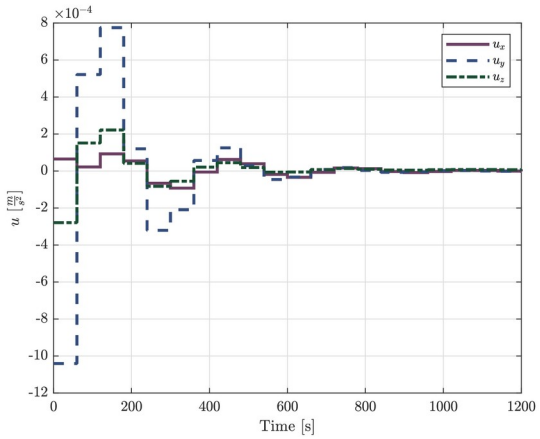

Figure 6: Control output delivered by the embedded execution.

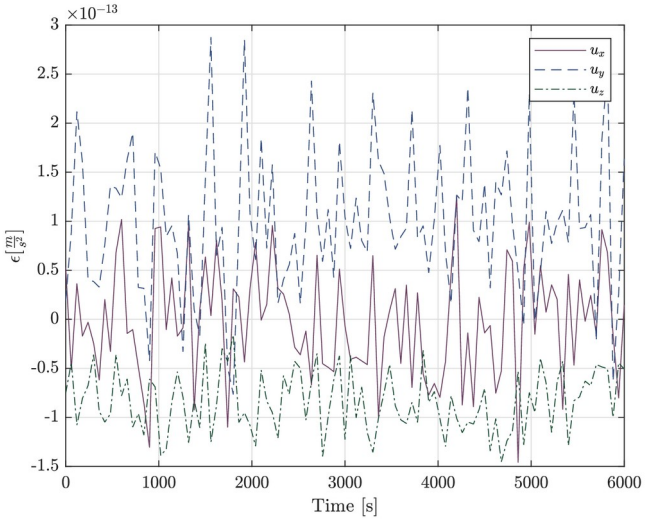

Figure 7: Discrepancy in calculation between PIL and MIL simulations for MBRL run in BeagleBone Black single-board computer.

Table 2: Average and maximum execution time of MBRL routines using a BeagleBone Black single-board computer.

Routine	Average [ms]	Max [ms]
ann()	0.89	0.90
mbrl.quadrpog()	191.10	272.20
mbrl.fmincon()	290.10	337.10

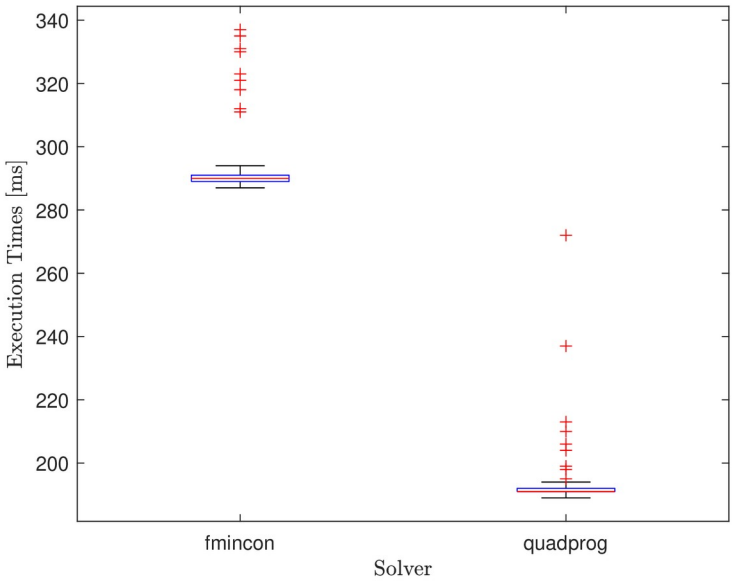

Figure 8: Statistics of execution times for the optimization solver used in the embedded application.

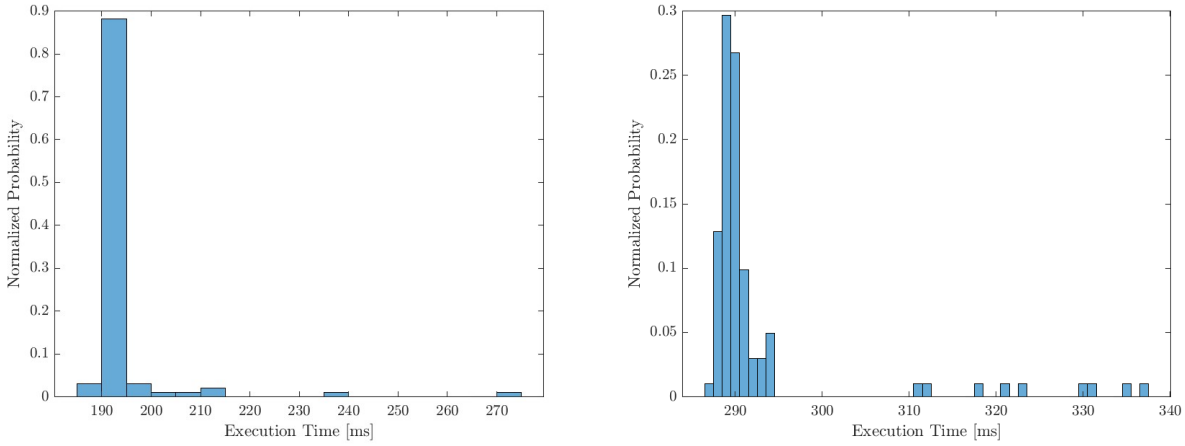

Figure 9: Normalized probability for the execution times profiled in the PIL simulation.From left to right, quadprog and fmincon are shown

Beside evaluating the computational times, it is critical to check whether the performed calculation are compatible with the results from the MIL simulations. Fig. 7 shows the discrepancy in the calculated control by the SBC and the Desktop simulator. The values are in the order of $\sim 10^{-13}$ ms^{-2} , which can be assumed to be an acceptable numerical error, thus validating the PIL test.

6. Conclusions

Considering execution times, task duty cycle and numerical discrepancy, the algorithms have been successfully validated in PIL tests. One of the major achievements of this work is the development of a complete pipeline to execute PIL tests for the developed algorithms.

The hardware presented in this paper is representative of the hardware integrated in the friction-less 5DoF facility installed at Politecnico di Milano [7][8][9]. Naturally, the algorithm development follows the same workflow adopted here: the numerical algorithms are coded in Matlab/Simulink ® , tested in Desktop computers and finally automatically ported into the target hardware using the dedicated coder. This enables fast prototyping and hardware validation of the selected algorithms.

The Neural Network Artificial Potential Field (NNAPF) algorithm has been tested numerically and in Processor-In-the-Loop (PIL) simulations using a single core TMS320C28x 32-Bit CPUs @200 MHz of TI C2000-Delfino MCUs F28379D unit. The numerical and PIL tests demonstrated:

- the neural reconstructed dynamics enables more accurate trajectory reconstruction and final target configuration with respect to Artificial Potential Field algorithm.
- the algorithm can be executed using limited computational power, thus making it suitable for on-board applications.

Moreover, the optimization-based algorithms have been tested and validated Processor-In-the-Loop (PIL) using a single-board computer BeagleBone Black. In particular, the execution times and resource utilization has been positively assessed to evaluate the feasibility of the embedded implementation.

References

[1] S. Silvestrini, M. Lavagna, "Neural-aided GNC Reconfiguration Algorithm for Distributed Space System: Development and PIL test", Advances in Space Research, 2021, doi:10.1016/j.asr.2020.12.014
[2] S. Silvestrini, M. Lavagna, "Spacecraft Formation Relative Trajectories Identification for Collision-Free Maneuvers using Neural-Reconstructed Dynamics", AAS/AIAA SciTech Forum, Orlando, 6-10 January 2020, doi:10.2514/6.2020-1918
[3] S. Silvestrini, M. Lavagna, "Processor-in-the-Loop Testing of AI-aided Al- gorithms for Spacecraft GNC", 71 st International Astronautical Congress, The Cyberspace Edition, virtual, 12-14 October 2020
[4] V. Pesce, S. Silvestrini, M. Lavagna, "Radial Basis Function Neural Network aided Adaptive Extended Kalman Filter for Spacecraft Relative Navigation", Aerospace Science and Technology, vol. 96, 2020, doi:10.1016/j.ast.2019.105527
[5] S. Silvestrini, M. Lavagna, "Inverse Reinforcement Learning for Collision Avoidance and Trajectory Prediction in Distributed Reconfigurations", 70 th International Astronautical Congress, Washington, USA, 21-25 October 2019
[6] A Pellacani, M Graziano, and M Suatoni, “Design , Development, Validation and Verification of GNC technologies”, in EUCASS2019, 2019. doi: 10.13009/EUCASS2019-38.
[7] P. Visconti, S. Silvestrini, and M. Lavagna, “Dance: a Frictionless 5 DOF Facility For GNC Proximity Maneuvering Experimental Testing And Validation”, in 69th International Astronautical Congress, 2018, pp. 1–5.
[8] D. Ottolina, S. Silvestrini, and M. Lavagna, “DANCE: Design and Characterization of a 5 DOF Facility for Relative GNC”, in ASTRA, 2019.
[9] S. Silvestrini, D. Ottolina, R. De Gasperin, M. Lavagna,“DANCE: Integration and Avionics Testing of 5 DOF Experimental Facility for Relative GNC”, in 71st International Astronautical Congress, 2020, pp. 1–5.