Model Predictive Control and Integration with the Autonomy Stack of the Astrobee Free-Flyer

Monica Ekal[1]*, Keenan Albee[2]*, Brian Coltin[3], Rodrigo Ventura[1], Richard Linares[2], and David W. Miller[2]

[1] Institute for Systems and Robotics, Instituto Superior Técnico [2] Department of Aeronautics and Astronautics, Massachusetts Institute of Technology [3] SGT Inc., NASA Ames Research Center

Introduction

Autonomous microgravity robots, like the Astrobee assistive free-flyer, must often interact with dynamic environments and systems with uncertain properties. The ability to perform receding horizon planning and control can enhance system performance and safety by determining control actions online using the latest available information. Future robotic free-flyers will use such algorithms to manipulate cargo, assemble on-orbit structures, and safely assist astronaut activities. The Astrobee platform will be a key enabler of research into this area from a guidance, navigation and control (GNC) and autonomy perspective. Two main contributions are detailed:

- Implementation and testing of the first model predictive controller on Astrobee hardware
- Software integration details of the algorithm and repurposing the Astrobee Flight Software (AFS) for guidance navigation and control (GNC) research, with accompanying software guide

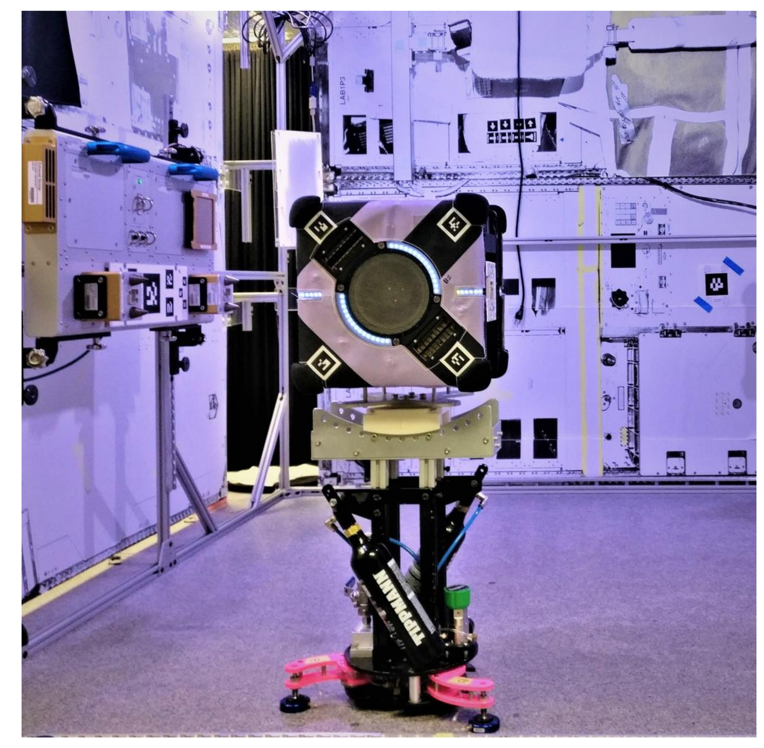

Figure 1: The Astrobee robotic free-flyer on an air bearing at the NASA Ames ground test facility during testing. Astrobee's capabilities as a GNC testbed are becoming evident.

GNC First Steps: Model Predictive Control

A standard model predictive control (MPC) receding horizon control algorithm was implemented for Astrobee's planar dynamics [1]. Shown below is the cost function J, which is optimized at each time step with weightings on state error (Q), input (R), and a terminal weight (H). MPC can add constraints arbitrarily, a key benefit over many other optimal controllers.

$$\underset{\mathbf{u_i}(-),\mathbf{x_i}(-)}{\text{minimize}} \quad J = \frac{1}{2}\sum_{i=0}^{N-1}[\mathbf{x}_i - \mathbf{x}_{i,des}]^\top \mathbf{Q}[\mathbf{x}_i - \mathbf{x}_{i,des}] + \frac{1}{2}[\mathbf{u_i} - \mathbf{u}_{i,des}]^\top \mathbf{R}[\mathbf{u}_i - \mathbf{u}_{i,des}] + \frac{1}{2}[\mathbf{x} - \mathbf{x}_{des}(N)]^\top \mathbf{H}[\mathbf{x} - \mathbf{x}_{des}(N)]$$

$$\text{subject to} \quad \mathbf{x}_{i+1} = f(\mathbf{x}_i, \mathbf{u}_i), \quad \mathbf{u}_{min} \le \mathbf{u}_i \le \mathbf{u}_{max}, \quad \mathbf{x}_i \in \mathcal{X}_{free}$$

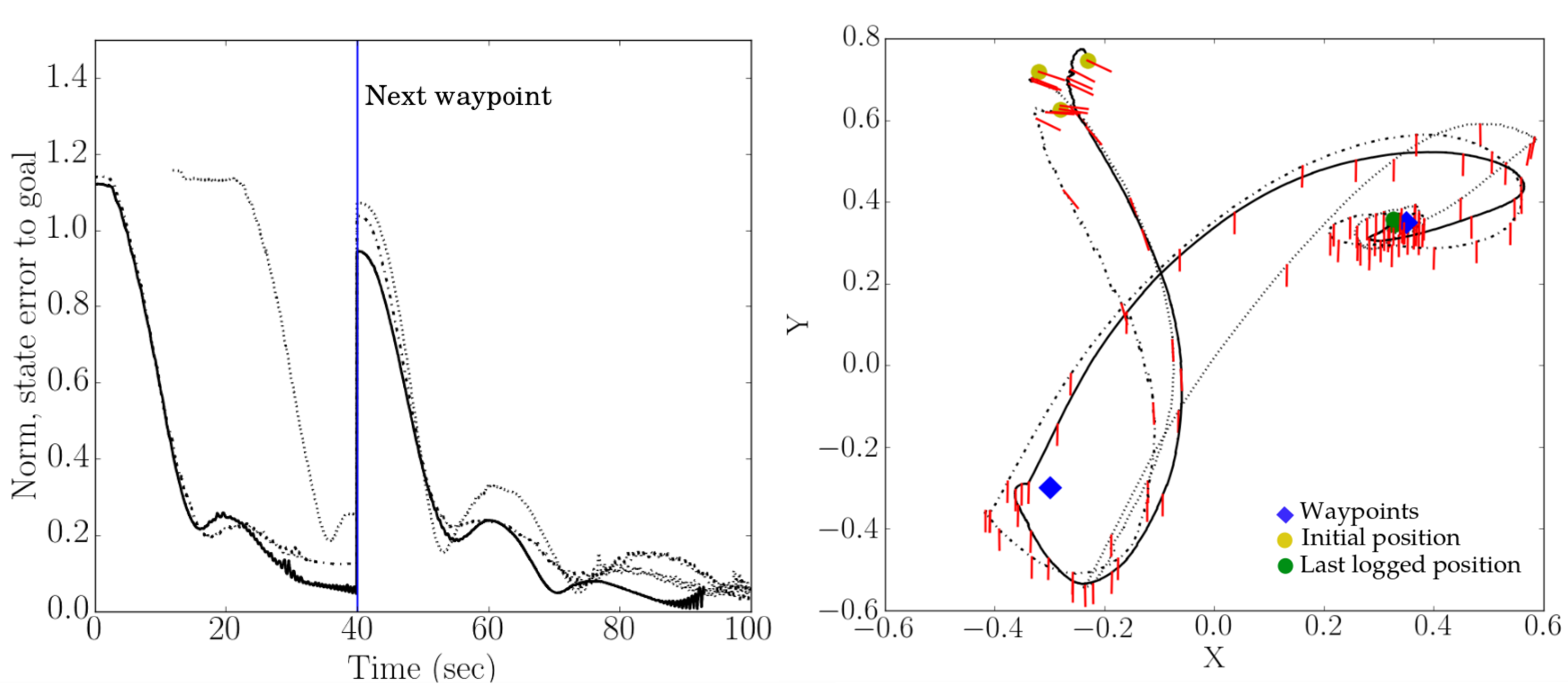

Figure 2: Normalized state error (left) with a waypoint change, and the path tracked by Astrobee (right) in the horizontal plane for three hardware runs. The red dash represents the pose of the x-axis of robot, plotted to show orientation.

Integration with Astrobee's Autonomy Stack

This work was one of the first uses of Astrobee as a GNC research testbed [2][3]. Many practical hurdles were overcome to integrate with Astrobee's autonomy stack. Moreover, these solutions are being provided to the broader Astrobee research community including [4]:

- Overriding default planning and control architecture;
- Providing build/test/document strategies for Astrobee development;
- Identifying Astrobee software interfaces for GNC/autonomy;
- Integrating outside libraries and GNC code (e.g., ACADO, CasADi).

Astrobee Autonomy Stack Highlights

Astrobee's Autonomy Stack consists of a high-level manager (Executive) which oversees various *ROS nodelets* that encapsulate key functionalities. A Choreographer and Planner coordinate building motion plans, while a GNC subsystem handles estimation (EKF), control (CTL), and the mixer (FAM). A pipeline to add custom GNC functionality to Astrobee was developed and is shared in detail in [4].

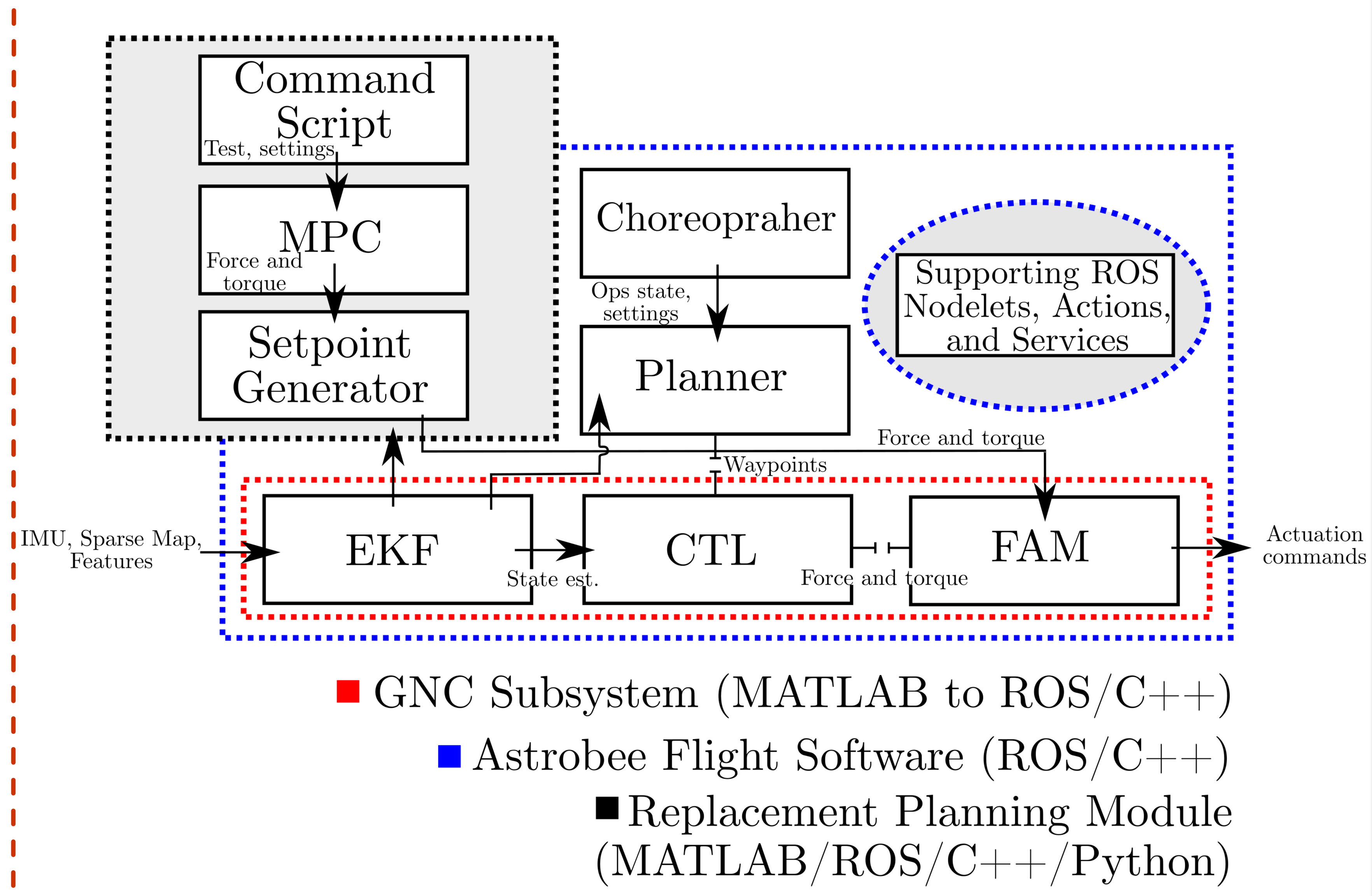

Figure 3: An overview of Astrobee's autonomy components relevant to GNC. The black outline shows the integrated command and control functionality, which overrides Astrobee's default Planner and CTL interfaces. A wide variety of integration schemes are possible, including code originally developed in MATLAB or higher-level languages when integrated in this way.

Conclusion

Initial integration tests of MPC were successfully carried out on Astrobee hardware. Further details on this implementation can be found in the technical report accompanying this poster. Over the course of this work, an in-depth understanding of Astrobee's software stack was gained, and important strides were made in shaping GNC integration which are shared in a software guide [4]. Future work will build on this platform to enable real-time microgravity probabilistic planning and estimation research.

References

[1] J. B. Rawlings, D. Q. Mayne, and M. M. Diehl, *Model predictive control: theory, computation, and design*, vol. 197. 2019.; [2] Smith, T. et al. (2016) Astrobee: A new platform for free-flying robotics on the international space station.; [3] Fluckiger, L. et. al. (2018). Astrobee robot software: a modern software system for space.; [4] K. Albee, M. Ekal, and C. Oestreich, "A Brief Guide to Astrobee's Flight Software," 2020. Available: https://github.com/albee/a-brief-guide-to-astrobee

Acknowledgements

This work was supported by the NASA Space Technology Mission Directorate through a NASA Space Technology Research Fellowship under grant 80NSSC17K0077, the LARSyS - FCT Plurianual funding 2020-2023, P2020 INFANTE project 10/SI/2016, and an MIT Seed Project under the MIT Portugal Program. Thank you to the Astrobee team at NASA Ames.

* Equal contribution

Contacts – mekal@isr.tecnico.ulisboa.pt, albee@mit.edu