Reconfigurable Onboard Guidance for (Micro-) Launchers

Malte Neuss [(1)], Dimitrios Gkoutzos[(2)], Tiago Milhano [(3)] , Said Jamal Mohamad [(4)]

[(1)]Astos Solutions, Meitnerstr. 8, 70563 Stuttgart, Germany, +49 711 892633 - 31, malte.neuss@astos.de
[(2)]Astos Solutions, Meitnerstr. 8, 70563 Stuttgart, Germany, +49 711 892633 – 27, dimitrios.gkoutzos@astos.de
[(3)]Astos Solutions, Meitnerstr. 8, 70563 Stuttgart, Germany, +49 711 892633 - 01, tiago.milhano@astos.de
[(4)]Astos Solutions, Meitnerstr. 8, 70563 Stuttgart, Germany, +49 711 892633 - 18, said.mohamad@astos.de

ABSTRACT

Accompanying the rising number of (micro-) launcher companies is an increasing demand for an 'off-the-shelf' Guidance, Navigation and Control software. In response, Astos is developing a Guidance and Navigation (GN) software that is reconfigurable to different launch vehicles and missions. This paper focuses on guidance and discusses some of the architectural and algorithmic design choices that facilitate its reconfiguration. To that end, an emphasis was placed on modularity and versatility both in the software architecture and the implementation of guidance algorithms. The onboard trajectory optimization algorithm proved particularly important in achieving a reconfigurable guidance and different methods to increase the range of manoeuvres and launch vehicles it can be applied to are discussed. In addition to the development of the GN software, a pipeline was created to automate its reconfiguration and thereby minimize the time and labour required by this process.

The reconfigurability of guidance is exemplified by applying the GN software to two different launch vehicles that are executing two distinct missions. Conducting a Monte Carlo test campaign for each scenario demonstrated that the guidance software can be reconfigured in a simple and timely manner while reliably and accurately attaining the target orbit.

1 INTRODUCTION

The rapidly growing and highly competitive nature of the commercial NewSpace market encourages its companies to cut costs and speed up development. However, emerging micro-launcher companies usually lack the expertise acquired from decades of experience or access to existing technology that can be utilized to accelerate the development of a new launch vehicle. Therefrom arises the need for industry tools that can compensate for these impediments. The design of the Guidance, Navigation and Control (GNC) software presents an expensive and time-consuming task in the development of a launch vehicle. An 'Off-The-Shelf' GNC software that is applicable to different stages in the design process, ranging from preliminary analyses up to detailed design, could accelerate the development tremendously. The primary requirement for such a GNC software is to be easily and rapidly reconfigurable to different launch vehicles and missions.

In the past, reconfigurability played a minor role in the design of GNC flight software, however, it has risen in importance as the variety and complexity of missions has increased with the development of larger and more versatile launch systems. While this has incentivised the development of GNC flight software that is reconfigurable to a specific set of missions the flight software usually remains tailored to a specific launch vehicle. Since the GNC flight software is typically developed by and for the same organisation developing the launch vehicle, there is little benefit in pursuing reconfigurability beyond the specific set of missions executed by their particular launch vehicle let alone design a GNC flight software that is applicable to different missions and launch systems.

To bridge the gap between the needs of (micro-) launcher companies and the lack of 'Off-The-Shelf' GNC flight software, Astos develops a reconfigurable guidance and navigation (GN) software up to Technology Readiness Level (TRL) 6 within the "Off-The-Shelf Guidance & Navigation for Microlauncher" (MLGN) activity. The MLGN project is carried out under and funded by the European Space Agency, as part of the Future Launchers Preparatory Programme. This paper focuses on guidance and outlines some of the developments and design choices that facilitate its rapid and simple reconfiguration. These considerations have affected the guidance software on both the architectural and algorithmic level but also led to the creation of a pipeline that automates the reconfiguration process. First, an overview of the guidance software architecture is provided before the process to reconfigure guidance is described in greater detail. Thereafter, the impact of reconfiguration on the algorithmic design of guidance is discussed. Lastly, the reconfiguration of the GN software is demonstrated by applying it to two distinct micro-launchers and missions and comparing the closed-loop simulation results to the respective offline optimizations.

2 GUIDANCE ARCHITECTURE

The guidance software architecture emphasizes modularity to facilitate the versatility and flexibility that is required by its application to various (micro-) launchers and missions. It is composed of a set of independent guidance modes, that cover the various functionalities (micro-) launchers require. Once a guidance mode is active it provides the attitude and throttle commands, hereafter referred to as guidance commands, that are executed by the launch vehicle. Examples of guidance modes include the endo-atmospheric ascent mode, which uses an open-loop table that is generated from an offline trajectory optimization, as well as a coasting, deployment and de-orbit mode. An important element of guidance is the closed-loop trajectory optimizer, which performs onboard trajectory optimizations. These are executed in closed-loop to correct errors that accumulated prior to or during the propulsive manoeuvre and are used to plan most propulsive phases. Subsequently, the closed-loop trajectory optimizer is a module that interacts with multiple guidance modes and must be versatile and robust enough to handle various types of manoeuvres as well as launch vehicles. The implications of these requirements on the choice of optimization algorithm and the design of guidance algorithms in general are discussed in greater detail in Section 4.

The top-level, modular architecture of guidance is shown in Figure 1. It has two main inputs, which are the navigation solution and the guidance specific commands from the Launch Vehicle Manager (LVM). These commands control the operation of guidance and specify the currently active guidance mode, its operating mode and the transition to the next guidance mode. Based upon these inputs, the active guidance mode will determine the guidance commands of the launch vehicle. This might entail a closed-loop trajectory optimization, depending on the active guidance mode. Once

the guidance commands have been determined, they pass through the 'Command Constraints' block which verifies and enacts constraints on the guidance commands, such as maximum attitude rates. Hereafter, these commands are output by guidance and passed to the control system for execution as well as the LVM for its next iteration.

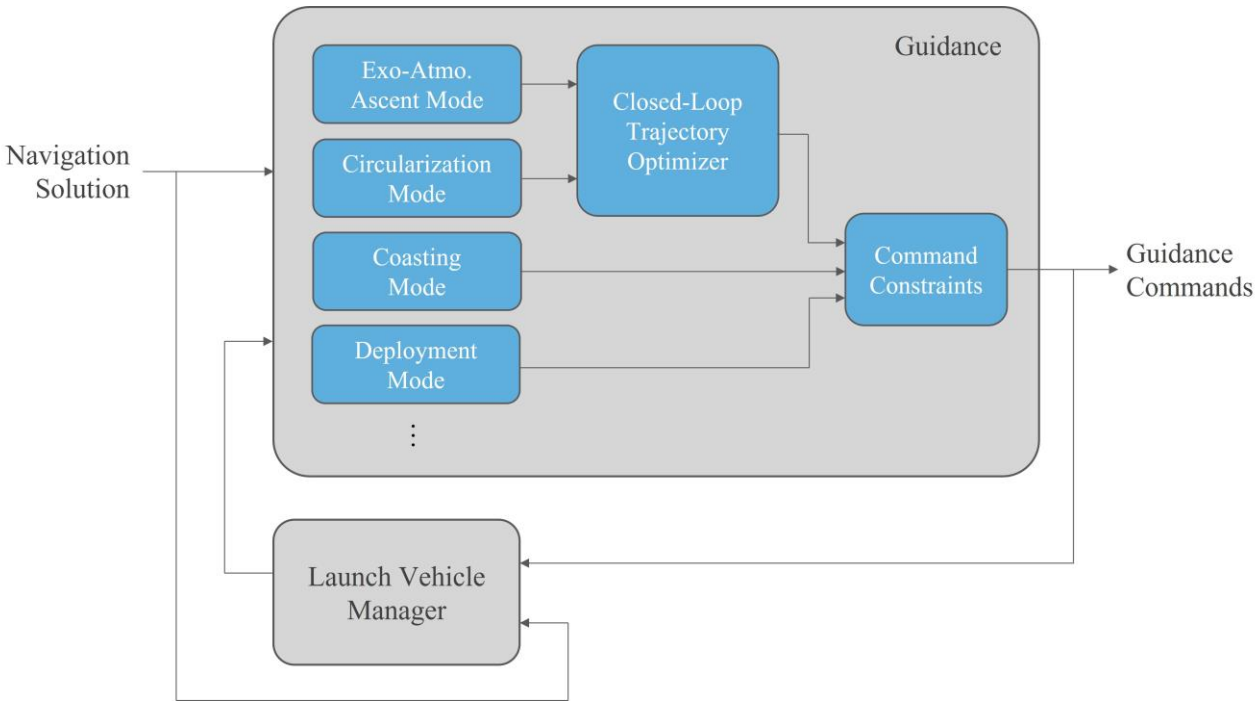

Figure 1. Top-level guidance software architecture.

The advantage of this modular architecture is that it allows any mission to be expressed as a sequence of guidance modes, provided that the guidance modes cover all the required functionalities. A further benefit is that new functionalities can easily be added by either creating new guidance modes or extending the capabilities of existing guidance modes. The LVM then executes the predefined sequence of guidance modes by controlling the transition from one guidance mode to the next. The command to transition to the next guidance mode is issued when a predefined condition is met. Common conditions include waiting for a fixed duration, waiting for a trigger signal or reaching a specific velocity. This ability to freely chain guidance modes and choose their transition conditions provides the flexibility and versatility to adapt guidance to different launch scenarios, where a launch scenario refers to a combination of launch vehicle and mission.

In addition to the flexibility generated by the guidance architecture, a successful reconfiguration of guidance necessitates an implementation of the guidance modes that encompasses all functionalities different launch scenarios might require. This requirement affects each guidance mode differently, with some requiring only modest modifications whilst others must accommodate a variety of applications. One such example is the coasting mode, which can have numerous applications such as adding a short coasting phase after a stage separation and the ignition of the next engine, coasting to a specific point along a transfer ellipse or performing a slew manoeuvre to the optimal attitude of the next propelled phase. Depending on the application, the methods to determine the coasting duration and attitude commands will differ and range from fixed values to more complex online estimations. While this versatility is necessary to realize the reconfiguration of guidance it has the downside of creating an overhead in code, computation time and memory when compared to a software tailored to a specific launch vehicle and mission. The impact of the overhead on the

overall computation time and memory consumption has been tested and it was shown that the GN software easily met our memory, CPU and real-time requirements. Furthermore, if computational resources are very scarce, once can remove the excess functionalities and create a flight software tailored to a particular launch scenario.

3 PIPELINE FOR GUIDANCE RECONFIGURATION

Offline trajectory optimizations can be used as a first estimate to judge the performance and behaviour of a launch vehicle executing a particular mission. However, this method considerably overestimates the actual performance since the offline trajectory optimization does not account for the numerous external disturbances and uncertainties in the launch vehicle's properties that affect an actual launch. A more realistic assessment of the launch scenario can be attained by simulating these variations and external perturbations in Model In the Loop (MIL) Monte Carlo simulations using the GNC software in closed-loop. To perform these efficiently for different launch vehicles or missions, it is not only paramount to have a reconfigurable GNC software but also a process that rapidly executes all steps necessary to move from an offline trajectory optimization to a MIL test. This process must acquire, analyse and synthesise all GNC relevant information into the parameters needed by the GNC software as well as setting up the MIL test environment. As part of the MLGN activity, a pipeline was developed to automate this process, the flow-chart of which is shown in Figure 2. Note that while the pipeline was developed with the entirety of the GNC software in mind, this paper presents the aspects relevant to guidance.

The first step is to set up the launch vehicle and mission in a trajectory optimization tool and optimizing the trajectory considering any relevant constraints. In this activity, the multi-purpose optimization software ASTOS was chosen for this purpose as well as the Dynamics, Kinematics and Environment model (DKE) for the Functional Engineering Simulator (FES) in the MIL or Software In the Loop (SIL) simulations. One of the benefits of using ASTOS for both the offline optimization and DKE model is that it simplifies the transition from the offline optimization to the DKE model and thus reconfiguration of the FES. In order to reconfigure the guidance software, first an export is generated from the offline optimization, which contains all the information needed by the guidance algorithm. This export is mostly composed of information regarding the launch vehicle, such as engine properties, dry and propellant masses, as well as mission specific information such as target orbits and attitude profiles used for open-loop tables. The required guidance export can be generated automatically using ASTOS, however, any tool can be used as long as it can supply the necessary information in the format specified by the guidance export.

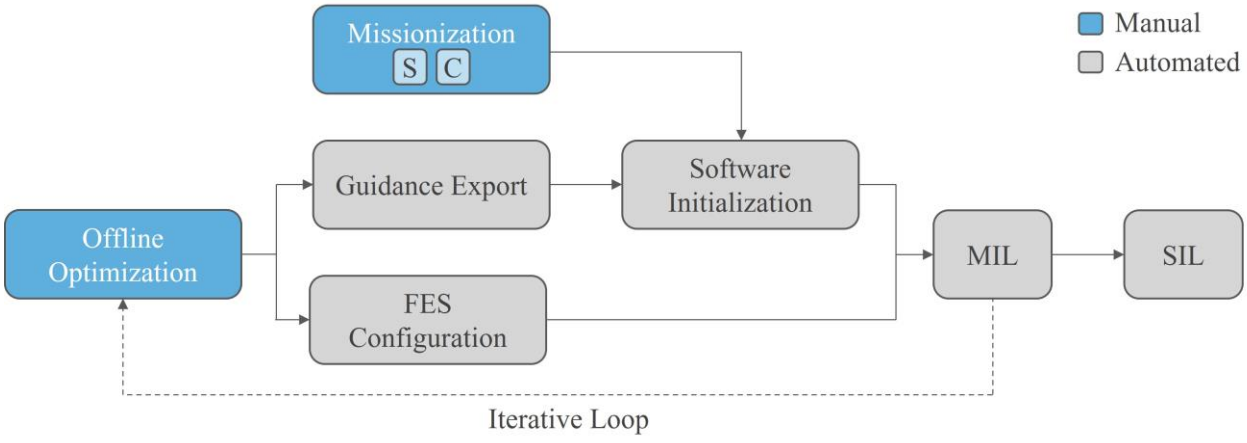

Figure 2. The pipeline used to reconfigure the guidance software. The Missionization step contains

‘Mission Set-Up’ (S) and ‘Guidance Configuration’ (C) files, which the user may need to alter as part of the reconfiguration process.

In the ‘Software Initialization’ all the parameters required by the guidance software are either loaded or generated. This process requires the launcher and mission specific information contained within the Guidance Export as well as guidance specific configuration parameters. These are separated into different files based upon their (in-)accessibility to the user and (in-)dependency on the Guidance Export. There are two files that are accessible to the user and these separate the information required to set-up the mission (‘Mission Set-Up’ (S)) from guidance configuration parameters (‘Guidance Configuration’ (C)) which possess default values. This structure promotes the rapid reconfiguration of guidance since it minimizes the information the user must manually supply while maintaining the ability to tune the behaviour of guidance. Within the ‘Mission Set-Up’ file, the user must specify the sequence of guidance modes that will be executed in the closed-loop simulation, the condition to transition to the next guidance mode and if required, the operating mode of the guidance mode. Optionally, one can further specify results from the missionization process, such as alternative target orbits for off-nominal scenarios like engine failures. The ‘Guidance Configuration’ file contains all tuneable guidance parameters that the user may modify to alter the behaviour of guidance. Examples include the operating frequency of guidance or the convergence criteria of the trajectory optimization algorithm. Most commonly, these parameters are used to either fine-tune the behaviour of guidance or experiment with different settings since the default values have been proven to work well for a variety of launch vehicles and missions, as will be demonstrated in Section 5. The information contained in these files is then used to select and process the relevant information from the Guidance Export into the remaining parameters required by the guidance software.

Once the initialization of the guidance software and configuration of the FES is complete, the MIL simulation can be executed. Using the Embedded Coder, source code can be generated and used for SIL tests. This process will be configured such that the final code has all parameters of the guidance software contained within one external file. This structure allows for changes in the launch scenario or ‘Mission Set-Up’ to be propagated to the SIL test through the replacement of the parameter file without having to generate or qualify the SIL code anew.

The development of a launch vehicle as well as its application to a specific mission are iterative processes and it is in these processes where the benefits of the reconfiguration pipeline become most apparent. The typical iterative loop consists of small changes to the launch scenario that is then reoptimized and tested with the guidance software in single or multiple MIL tests to gain a realistic insight into the launch vehicle’s performance and behaviour. When developing a launch vehicle, these changes are typically variations in the launch vehicle’s properties, such as dry and propellant masses or engine performance. On the other hand, applying a launch vehicle to a specific mission also requires iterative testing in order to determine mission specific parameters such as the maximum payload that can reliably achieve the target orbit or the recovery orbits that can be targeted in case of off-nominal occurrences.

The reconfiguration pipeline outlined in this paper accelerates this iterative loop considerably by automating most of the steps between the offline optimization and the MIL tests. The most labour-intensive steps are the initial set-up of the launch scenario in the offline optimization tool and specifying the ‘Mission Set-Up’. Once these are completed, the iterative loop requires minimal manual input. For instance, when developing a launch vehicle by tweaking engine properties for instance, the user must only modify and re-optimize the offline optimization scenario since the ‘Mission Set-Up’ usually remains unchanged. The same is true when determining the maximum

payload mass for a particular mission. Changing optional parameters such as recovery orbits or the operating mode of a guidance mode in the 'Mission Set-Up' requires even less labour since the offline optimization scenario does not need to be altered or reoptimized. These examples demonstrate the tremendous reduction in time and labour that is achieved by using the reconfiguration pipeline in the iterative development or application of a launch vehicle.

4 ALGORITHMIC DESIGN OF GUIDANCE

The closed-loop trajectory optimizer is the centrepiece of the guidance software as it plans the majority of exo-atmospheric, propulsive manoeuvres and interacts with multiple guidance modes. Subsequently, the breadth of launch scenarios the guidance software can be reconfigured to directly depend on the spectrum of launch vehicles and manoeuvres the closed-loop trajectory optimizer can be applied to. With that in mind, the Powered Explicit Guidance (PEG) algorithm developed by NASA [1] was chosen as the closed-loop trajectory optimizer. Its extensive flight heritage which includes the Space Shuttle Program, the Space Launch System (SLS) [2] and Orion spacecraft [3], demonstrates its adaptability to different missions and launch vehicles. In combination with its modular architecture and low computational load, it is a sound choice for a reconfigurable on-board trajectory optimizer.

Algorithms designed for on-board trajectory optimization often solve a simplified optimization problem to reduce computational loads and improve their robustness to initial conditions and variations in launcher properties. However, these simplifications often restrict the type of launch vehicles and manoeuvres to which the optimization method can be applied. For instance, PEG estimates the impact of gravity on its propulsive trajectory using a Keplerian orbit propagation and this simplification is one of the reasons why the accuracy and robustness of PEG diminishes for long, low thrust manoeuvres [2].

It is important to extend the capability of the on-board trajectory optimization in order to maximize the variety of launch scenarios that the guidance software can be reconfigured to. This can be achieved through a variety of methods, one of which is to develop an existing algorithm to expand its applicable range. For example, NASA developed updates to the PEG algorithm for its SLS Block-1 launch vehicle to make it more robust for low thrust manoeuvres [2]. Alternatively, one can improve the estimate of the gravitational influence on the trajectory by replacing the Keplerian orbit propagator with a more advanced alternative. Apart from improving the performance of the optimization method one can try to extend its capabilities by including additional parameters such as thrust profiles or throttling. However, while these modifications expand the range of launch scenarios that the optimization algorithm may be applied to, it does not overcome the fundamental limitations of the algorithm since these arise from its underlying assumptions. Subsequently, an alternative approach is not to use a single optimization algorithm but to establish a library of complementing on-board trajectory optimization algorithms and selecting the method most suited to the specific launch vehicle and manoeuvre.

While the first two approaches only affect the trajectory optimizer, the rest of guidance can also contribute to making the on-board trajectory optimization more robust. For example, some of the inputs to the on-board trajectory optimization, such as the total mass of the launch vehicle, need to be estimated during flight. Providing more accurate estimates will improve the performance of the optimized solution as well as the reliability with which a solution is found. Similarly, performing a slew manoeuvre to the optimal starting attitude of the next propulsive phase will save propellant and improve convergence during the closed-loop trajectory optimization due to preferable starting

conditions.

The robustness can further be improved by minimizing the error in the initial state of the launch vehicle with respect to the offline optimization. This is particularly important when a mission consists of multiple propulsive phases since the error in the initial target orbit will impact the trajectory of the subsequent manoeuvre. For example, consider a mission in which a launch vehicle launches into the perigee of a transfer ellipse and coasts to the apogee where the orbit is circularized. Small errors in the injection velocity at the perigee can substantially raise or lower the apogee, which then requires large corrections during the circularization burn. This increases the complexity and duration of the circularization burn which in turn increases the difficulty of the trajectory optimization. This is further exacerbated by the fact that the optimal location on the transfer ellipse to start the circularization burn will have shifted. Using the optimal coasting duration or location from the offline trajectory optimization will initiate the manoeuvre at a sub-optimal point and further elongate the burn. One method to determine the optimal coasting duration is to execute the closed-loop trajectory optimization in the background and coast until the estimated burn duration reaches its minimum. One can avoid these continuous computations by finding a heuristic to estimate the optimal coasting duration based on the error in the injection into the transfer ellipse. While this strategy improves the robustness of the trajectory optimization process, it is usually more effective to reduce the error in the injection into the transfer ellipse in the first place. Since the orbital properties are very sensitive to the injection velocity, the error in the target orbit is directly related to the precision with which the Engine Cut-Off (ECO) can occur. This precision is ultimately dependent on the frequency at which the ECO can be commanded (f_{ECO}) and executed and, assuming a perfect navigation, results in a velocity error shown in Eq. 1,

$$-\frac{a(t_{ECO})}{2f_{ECO}} < v(t_{ECO}) - v_{target} \leq \frac{a(t_{ECO})}{2f_{ECO}} \qquad \text{Eq. 1}$$

where $v(t_{ECO})$ and $a(t_{ECO})$ are the speed and acceleration magnitude of the launch vehicle at ECO and v_{target} is the targeted speed for injecting into the target orbit. The frequency at which guidance operates is usually not high enough to provide an acceptable injection error, which is why the ECO scheduling occurs in a separate function operating at a higher frequency. The exact value of f_{ECO} will depend on the acceleration with which the launch vehicle injects into the target orbit as well as the requirements set on the injection accuracy and is thus specific to each launch scenario.

These are some of the code considerations that have been explored and implemented to make the trajectory optimization more robust and versatile. The resultant breadth of launch scenarios that the guidance software can be successfully applied to is demonstrated in Sec. 5 using the example of two distinctive launch scenarios.

5 APPLICATION OF GN SOFTWARE

The reconfigurability of the guidance software is demonstrated by applying it to two representative micro-launcher scenarios that differ both in their launch vehicle and mission. The first launch scenario (A) consists of a two-stage launch vehicle that directly targets a circular Sun-Synchronous Orbit (SSO) of 300km altitude, while the second (B) considers a three-stage launch vehicle that first enters a transfer ellipse before circularizing its orbit to reach a 500km high SSO. A more detailed overview of the different missions is provided in Tables 1 and 2.

This test was executed by creating launch scenario A in ASTOS and specifying the required information in the 'Mission Set-Up' file. The sequence of guidance modes executed by the guidance software is one of these required inputs and shown for each launch scenario in their respective table. The brackets next to the guidance mode further indicate whether the guidance mode uses an open-loop table (OL) or performs a closed-loop, onboard, trajectory optimization (CL). Once the information has been provided, the pipeline discussed in Section 3 automatically initializes the guidance software and the MIL tests can be executed. Hereafter, the reconfiguration process is repeated for launch scenario B. Note, that for all tests in both launch scenarios the same, default configuration parameters will be used so that only the Guidance Export, resulting from the offline optimization, and 'Mission Set-Up' differ between the launch scenarios.

Table 1: The mission timeline of launch scenario A and the corresponding sequence of guidance modes used in the MIL simulations.

Flight Time (s)	Event	Guidance Mode
0	Lift-Off	Endo-Atmospheric Ascent Mode (OL)
192	First Stage Burn Out,	
202	First Stage Separation, Start of Second Stage Burn	Exo-Atmospheric Ascent Mode (CL)
218	Fairing Jettison	
519	Payload Deployment	Deployment Mode
619	Deorbit Burn	Deorbit Mode (OL)

Table 2: The mission timeline of launch scenario B and the corresponding sequence of guidance modes used in the MIL simulations.

Flight Time (s)	Event	Guidance Mode
0	Lift-Off	Endo-Atmospheric Ascent Mode (OL)
127	First Stage Burn Out	
147	First Stage Separation, Start of Second Stage Burn	
238	Fairing Jettison	Exo-Atmospheric Ascent Mode (CL)
272	Second Stage Burn Out, Start of Third Stage Burn	
383	Start Coasting in Transfer Ellipse	Coasting Mode
1979	Circularization Burn Commences	Circularization Mode (CL)
2412	Payload Deployment	Deployment Mode
2512	Deorbit Burn	Deorbit Mode (OL)

The MIL tests consist of a closed-loop, 3 Degree of Freedom (DoF) Simulink simulation of the Guidance and Navigation (GN) software. The navigation consists of a hybrid (IMU/GNSS) extended Kalman Filter in the error state formulation, which was also developed as part of the MLGN activity. It was included in the tests to provide more realistic results and it was reconfigured in the same process as the guidance software. For each launch scenario, two tests were performed. The first is a 'nominal' test, which uses the same environmental properties, such as atmospheric density and temperature, and launch vehicle properties, such as dry masses, engine mass-flow rates

and nozzle areas, as the offline optimization. The second consists of 200 Monte Carlo simulations in which these properties are varied and external disturbances, such as wind, are added. Both the nominal and Monte Carlo tests include error modelling for the sensors used by navigation.

The attitude profiles that result from these tests for launch scenarios A and B, are shown in Figure 3 and Figure 4 respectively. In Figure 3, one can clearly identify the switch from the open-loop profile to the closed-loop phase 202 seconds after launch. The small magnitude of the attitude deviations between the offline trajectory optimization and nominal test demonstrate the effectiveness of PEG in determining the optimal trajectory onboard. On the other hand, the great range in attitude commands that occur during the Monte Carlo tests reveal the large errors that can accumulate during the open-loop phase and emphasizes the importance of having the closed-loop phase to correct these errors.

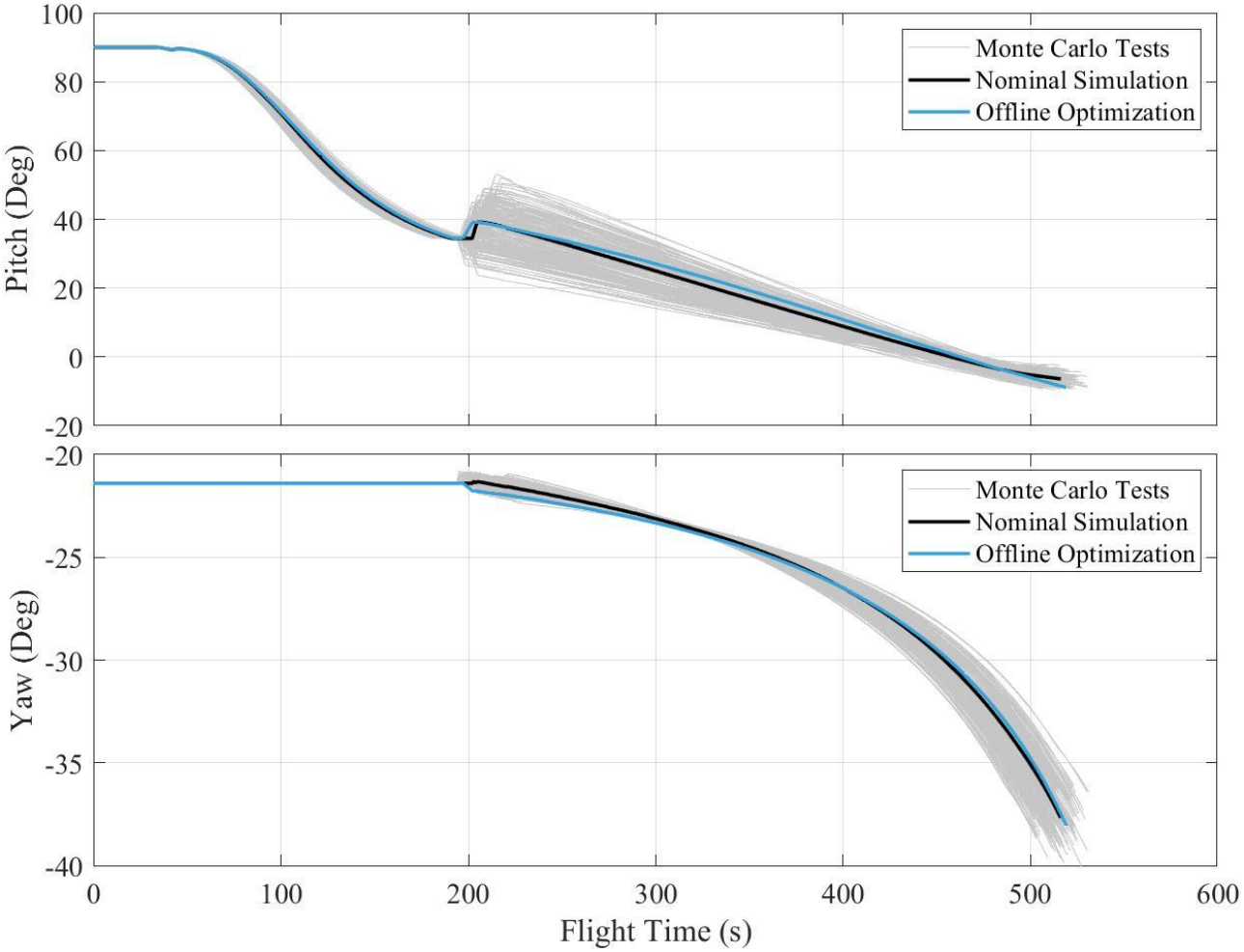

Figure 3: The attitude profile of launch scenario A. The offline ASTOS optimization (blue) is compared to a nominal, closed-loop, 3-DoF Simulink simulation of the GN software (black) and 200 Monte Carlo Simulations (grey)

The same trends can be observed in Figure 4, where the first closed-loop phase occurs 238 seconds after launch. In this launch scenario one can observe a greater difference between the solution found by PEG and ASTOS, however, this has no impact on the overall performance since the burn duration of the nominal test and offline optimization is the same. In the subsequent coasting phase, the attitude profiles differ between the offline optimization and the MIL tests, since the offline

optimization and guidance software use different interpolation strategies. The offline optimization linearly interpolates towards to the initial attitude of the circularization burn over the entire coasting trajectory. This is not possible for the guidance software since the coasting duration and optimal starting attitude of the circularization burn are initially not known. Subsequently, the guidance software interpolates at a predefined angular rate towards the initial attitude of the circularization burn of the offline optimization and performs a slew manoeuvre to the attitude determined by the onboard optimization just prior to ignition.

The circularization burn commences about 2000 seconds after launch. The difference between the nominal test and offline optimization stems from the error in the injection into the transfer ellipse. This error falls randomly within the range specified by the sum of Eq. 1 and the error in the velocity estimate of navigation and thereby raises or lowers the apogee beyond the circular target orbit. These deviations need to be corrected within the circularization burn, causing the launch vehicle to pitch up or down to raise or lower the apogee. This behaviour can be seen in the Monte Carlo tests where the pitch corrections are uniformly distributed around the offline optimization. One can further see that the circularization burn begins at different times for different Monte Carlo simulations, in order to minimize the duration of the circularization burn. The launch vehicle performing this manoeuvre has a low thrust-to-weight ratio, which results in an acceleration due to thrust of about 0.3 m/s^2 at the injection into the target orbit. This produces a circularization burn lasting approximately 450 seconds and amplifies the magnitude of the attitude commands needed to correct the errors accumulated along the transfer ellipse and attain the target orbit. Since the performance of PEG diminishes for long, low-thrust manoeuvres, this demonstrates the effectiveness of the methods outlined in Section 4 in extending the range of applicable launch scenarios of PEG.

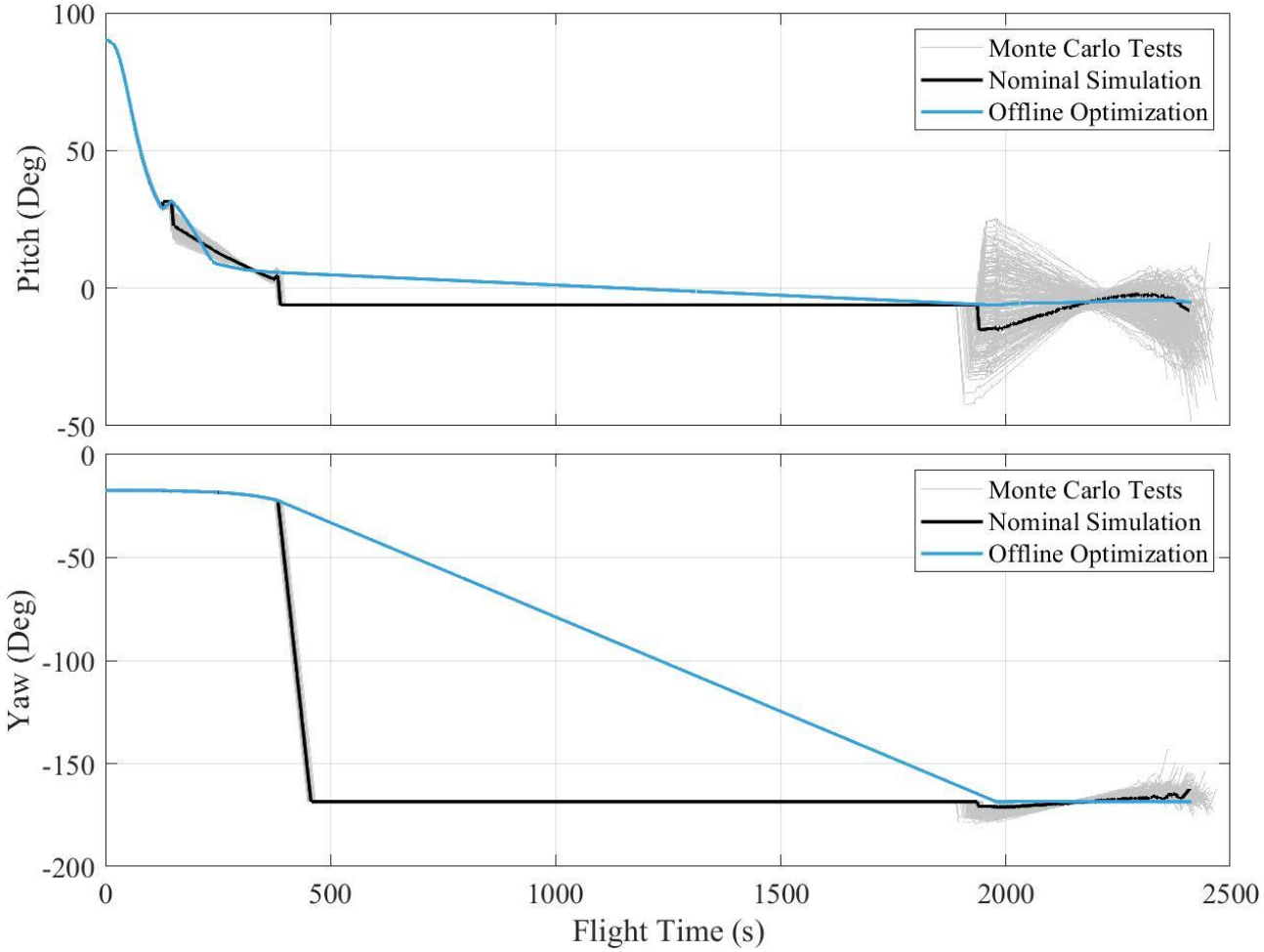

Figure 4: The attitude profile of launch scenario B. The offline ASTOS optimization (blue) is compared to a nominal, closed-loop, 3-DoF Simulink simulation of the GN software (black) and 200 Monte Carlo Simulations (grey).

The accuracy with which each launch scenario achieves its target orbit in terms of the semi-major axis (sma), eccentricity (e), inclination (i) and longitude of ascending node (Ω) at the time or orbit injection (t_i) is summarized in Table 3. The error in each orbital element is provided for the nominal simulation whilst the mean error (μ) and its standard deviation (σ) is shown for the Monte Carlo tests. The results demonstrate that both launch scenarios successfully and reliably reached their respective target orbits for both the nominal and Monte Carlo tests. As a reference, the three sigma requirements of the Vega launch vehicle [4] have been added to the table and it is evident that all but the eccentricity of launch scenario A, easily satisfy these requirements. However, it should be noted that these tests where not conducted in a 6 DoF environment and did not include control, the inclusion of which will worsen the accuracies and reduce the margins to the Vega requirements.

Table 3: The accuracy in the orbital parameters and acceleration due to thrust at the injection into the target orbit. For reference, the 3-sigma injection requirements of Vega are included [4].

	Launch Scenario A			**Launch Scenario B**			**Vega**
	Nominal	MC Tests		Nominal	MC Tests		
	-	μ	3σ	-	μ	3σ	3σ
sma(t_i) – sma$_{target}$ [km]	1.6	1.1	3.6	-0.13	-0.15	1.56	15
e(t_i) – e$_{target}$ ×1e-4	6.5	7.4	5.1	0.18	0.67	1.35	12

$\mathbf{i(t_i) - i_{target}}$ ×1e-3 [deg]	-2.2	-3.0	1.1	0.54	0.17	1.89	150
$\mathbf{\Omega(t_i) - \Omega_{target}}$ ×1e-2 [deg]	-1.5	-1.8	0.6	0.025	0.007	0.072	20
$\mathbf{a_T(t_i)}$ [m/s^2]	70			0.3			~1.0

The launch scenarios were also chosen to demonstrate the large range of thrust-to-weight ratios that guidance can be applied to and their effects on the injection accuracy of the target orbit. Since launch scenario A directly injects into the target orbit with its second stage, it does so with a large acceleration due to thrust of 70 m/s^2. In contrast, launch scenario B injects into a transfer orbit and uses a dedicated engine to perform the long, low-thrust circularization burn at an acceleration due to thrust of 0.3 m/s^2. Since the ECO occurs at the same frequency for both, the lower thrust-to-weight ratio of launch scenario B will make its injection into the target orbit much more precise than that of A, as can be seen in Table 3. By tuning the guidance parameters such as the convergence criteria of the trajectory optimization or the frequency of the ECO schedular one can improve the accuracy in the orbital elements, however, there is a limit to these improvements and they cannot bridge the gap in accuracy between launch scenario A and B.

Therefore, the precision and accuracy with which a target orbit can be achieved depend on the launch vehicle design and incentivises a final stage with a low thrust-to-weight ratio. This is reflected by the design of Vega, which uses the Attitude Vernier Upper Module (AVUM) to reach its target orbits. This module is located at the low end of the thrust-to-weight spectrum with an acceleration due to thrust of approximately 1 m/s^2 [4]. This large difference in acceleration between Vega and launch scenario A explains why the launch scenario cannot meet Vega's eccentricity requirement and cannot be expected to do so. On the other hand, launch scenario B should meet Vega's requirements and that can be seen in the results in Table 3.

Overall, the tests have shown that the guidance software can be successfully reconfigured in a simple and timely manner to launch scenarios that cover a wide range of thrust-to-weight ratios. The Monte Carlo tests further demonstrated that the target orbits are achieved accurately and reliably for both launch scenarios and highlighted the dependency of orbital injection accuracies on the launch vehicle design.

6 CONCLUSION

An 'Off-The-Shelf' GNC software that is applicable to different stages in the design process of a launch vehicle can help emerging micro-launcher companies to cut costs and accelerate development. In the MLGN activity, Astos developed a reconfigurable GN software to TRL 6 that can be integrated into flight software and has completed Processor-in-the-Loop tests. This paper focused on the guidance aspect of the GN software and discussed some of the architectural and algorithmic design choices that aide the reconfiguration of guidance, as well as the pipeline that was implemented to automate the reconfiguration process.

The software architecture of guidance supports its reconfiguration by introducing modular guidance modes which cover the various functionalities (micro-) launchers might require. These can be freely chained into any sequence and in combination with the ability to choose the condition that controls

the transition between guidance modes, this provides the flexibility to execute an extensive variety of missions. To fully exercise this flexibility, it is further required that each guidance mode encompasses all the functionalities that different launch scenarios might utilize. To reconfigure the guidance software in a simple and timely manner, a pipeline was developed that automatically reconfigures and re-initializes the guidance software based upon changes in the offline optimization or missionization of the launch scenario. This pipeline has proven particularly effective in reducing the time and labour required by iterative processes such as launch vehicle design and missionization. Lastly, different methods were discussed to increase the variety of launch scenarios to which the onboard trajectory optimization can be applied. These include the development of the existing trajectory optimization algorithm, the use of multiple trajectory optimization algorithms as well as changes to the guidance modes and the introduction of the ECO schedular which increase the robustness and accuracy of the system as a whole. Since the versatility of the guidance software is directly linked to the versatility of the trajectory optimization algorithm, these developments proved vital in achieving the goal of a reconfigurable guidance software.

The guidance and navigation software was applied to two launch scenarios that differ both in their launch vehicle and mission. The reconfiguration of the software only consisted of a change in the offline optimization and missionization parameters since the same, default configuration parameters were used for both launch vehicles. The Monte Carlo tests demonstrated that the target orbits are achieved accurately and reliably for both launch scenarios and further highlighted the dependency of orbital injection accuracies on the launch vehicle design. Overall, the tests have shown that the guidance software can be successfully reconfigured in a simple and timely manner to launch scenarios that cover a wide range of thrust-to-weight ratios while providing an accurate and performant solution that is robust to external perturbations and uncertainties.

7 REFERENCES

[1] Jaggers, R. F. *An Explicit Solution To The Atmospheric Powered Flight Guidance And Trajectory Optimization Problem For Rocket Propelled Vehicles*, AIAA Conference, Hollywood, Florida, 1977.

[2] Von der Porten P., et al. *Powered Explicit Guidance Modifications & Enhancements for Space Launch System Block-1 and Block-1B Vehicles*, NASA Technical Reports Server, 2018.

[3] Fill, T., et al. *Orion's Powered Flight Guidance Burn Options for Near Term Exploration Missions*, NASA Technical Reports Server, 2018.

[4] Arianespace *Vega User's Manual*, Issue 4, Revision 0, 2014.