Mission-Dependent Sequential Simulation for Modeling and Trajectory Visualization of Reusable Launch Vehicles

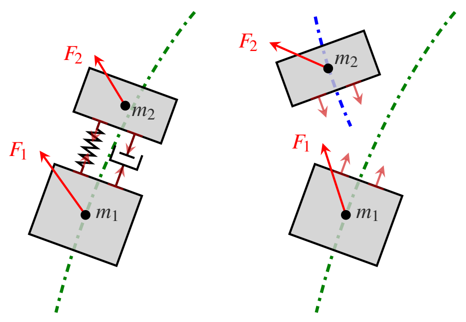
Lâle Evrim Briese

Institute of System Dynamics and Control, DLR German Aerospace Center, Oberpfaffenhofen, Germany,
`Lale.Briese@dlr.de`

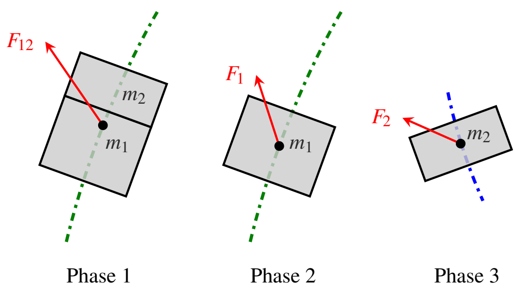
Abstract

The multibody modeling and visualization of reusable launch vehicles is a challenging task due to their variable structure regarding component separation and engine cutoffs during ascent and descent. However, the number of states within a MODELICA-based multibody model has to remain constant during a simulation. Therefore, the variable structure of launch vehicle models is often considered by using time- and state-dependent conditional statements and separation components. Such an approach can lead to a higher number of equations in the model and to a higher model complexity, respectively. In this paper, a mission-dependent sequential simulation approach for the modeling and trajectory visualization of launch vehicle systems is introduced. Here, the system is divided into characteristic phases, which are modeled with the DLR LauncherApplications Library capitalizing its modular, reusable and user-friendly structure to maintain compatibility between phases and to decrease the overall model complexity and the number of equations.

Keywords: launch vehicle modeling, trajectory, visualization, mission-dependent modeling, sequential simulation



(a) End-to-End Simulation.



(b) Sequential Simulation.

Figure 1. Schematical Representation of the Main Differences between End-to-End Simulation and Sequential Simulation.

1 Introduction

The multibody modeling of reusable launch vehicles can be a challenging task due to multiple disciplines involved in the modeling and simulation process, such as environment, aerodynamics, propulsion, structural dynamics, separation dynamics, as well as Guidance, Navigation and Control (GNC).

Especially, the variable structure of the overall multidisciplinary launch vehicle system has to be taken into account during the multibody modeling approach. Depending on the launch vehicle design and its mission requirements, the launch vehicle configuration can experience significant changes in its system structure. This can include the separation of stages, fairing, and payload as well as the time- and state-dependent *main engine cutoff* (MECO). These characteristic events for launch vehicles often lead to a change in the overall model structure and to a high number of states and equations due to separation or MECO as well as to non differentiable time-dependent step commands during engine ignition or cutoff.

In the object-oriented modeling language MODELICA (Modelica Association, 2014) the number of states and equations has to remain constant during a simulation. In recent developments, launch vehicle multibody models were obtained which use time- and state-dependent conditional statements to characterize the variable system structure, as presented in (Acquatella B., 2016). Also, separation models as introduced in (Acquatella B. and Reiner, 2014) can be used to fulfill the requirements for a fixed number of states and equations during the overall simulation by defining a connectivity condition between separated components and continuously simulating all components even after separation. These kind of models and methods are used within end-to-end simulations, which consider the multibody dynamics of all separated and connected launch vehicle components simultaneously as shown in Figure 1(a).

However, this approach for end-to-end simulations can lead to higher model complexity, number of equations and processing times. For instance, the separation models double the number of equations of motion since the generalized forces for each body have to be calculated, whether connected or separated, producing an overhead of unnecessary simulation data and calculations. Additionally, time-dependent conditional statements can result in time and state events, which can be undesired due to their disadvantages during the integration process.

As a consequence, new modeling and simulation methods are needed, which can reduce the model complexity and avoid an overhead of equations. Several methods have been developed over the past years for robust and efficient handling of variable structure systems with index changes as described for example in (Mehlhase et al., 2014; Mehlhase, 2015), (Zimmer, 2010) and (Elmqvist et al., 2014; Mattsson et al., 2015). These methods include a new experimental modeling language with an interpreter for variable structure DAE systems as proposed in (Zimmer, 2010), the usage of a *multi-mode Pantelides algorithm* to enable the simulation of models similar to state machines as shown in (Elmqvist et al., 2014; Mattsson et al., 2015) or the usage of complementary software languages and tools such as PYTHON or MATLAB for iterative simulations restricted by user-defined events.

Within this paper, a script-based sequential simulation method is proposed using only DYMOLA's built-in commands without the need for conventional variable structure modeling techniques or the usage of multiple software languages. The sequential simulation method shown in this paper is used especially for the modeling and visualization of launch vehicle systems, but can be implemented in any other application field. The modeling strategy is based on a similar approach used for multi-phase trajectory optimization of launch vehicle systems as described by (Schnepper, 2014). Therefore, the overall simulation is divided into characteristic *phases* or *modes* as referred to by variable structure modeling methods. These phases are then simulated sequentially while using consistent models with reduced complexity corresponding to each phase as shown in Figure 1(b).

First, a brief overview of the launch vehicle modeling framework is presented in Section 2. In Section 3 the sequential simulation method is introduced while taking a closer look into the multi-phase models used within trajectory optimization and visualization. In Section 4, the results regarding two launch vehicle concepts obtained by the proposed methods are presented. Finally, in Section 5 a summary and outlook will be provided.

2 Launch Vehicle Modeling

The launch vehicle modeling framework used for the modeling and visualization of expendable and reusable launch vehicles provides generic model components for launch vehicle simulations considering three or six degrees of freedom (DoF). Complementary to the *Modelica Standard Library* (Modelica Association, 2014) the following MODELICA-based libraries are used:

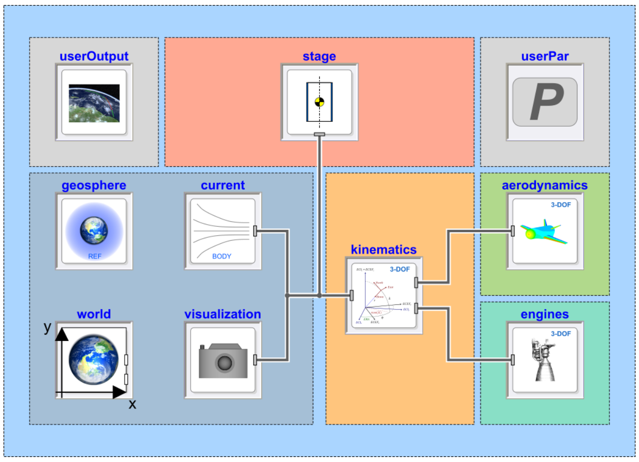


Figure 2. Overview of the 3-DoF / 6-DoF Launch Vehicle Modeling Framework provided by the DLR LauncherApplications Library.

- **DLR Environment Library** for modeling of environmental effects such as gravity acceleration of a rotating non-spherical planet (Briese et al., 2017).
- **DLR SpaceSystems Library** for modeling of satellite (Reiner and Bals, 2014) and launch vehicle systems (Acquatella B., 2016).
- **DLR LauncherApplications Library** for modeling of launch vehicle systems for example as consistent 3-DoF multibody models (Briese et al., 2018).

In Figure 2, a basic overview of the modular and internally consistent model structure of the modeling framework for a 3-DoF launch vehicle is given. All components are declared as `replaceable` models, which depend on dedicated and mutually shared *BaseClass* partial models as conceptually introduced in the DLR Environment Library. Since all components are based on the same parametric dataset *userPar*, this approach leads to a consistent model behaviour even if the level of detail is changed between simulations to conduct conceptual or detailed analyses assuming existent transition conditions.

All multibody related components are connected using multibody frames as defined in the *Mechanics.MultiBody* package of the *Modelica Standard Library* capitalizing the acausal structure of MODELICA. Mutually shared variables between main components are declared as input parameters and accessed by the parameter interface instead of using signal-based connectors to avoid using generic signal-based interfaces and thus unclear unit definition.

Within the launch vehicle modeling framework as shown in Figure 2 the *world* component provides the basic planet-dependent coordinate systems as well as more accurate gravity acceleration models. The *geosphere* component deliveres atmospheric parameters based on planet-specific atmosphere models. Since only the 3-DoF representation of the launch vehicle model for trajectory optimization is considered here, the usage of the *current* (wind) component will not be explained in this paper.

Table 1. Subset of the Basic Parameter Dataset *userPar*.

	Name	*Description*
3-DoF	start_lat	Initial Latitude
	start_lon	Initial Longitude
	start_h	Initial Altitude
	start_vel	Initial Velocity
	start_fpa	Initial Flight Path Angle
	start_chi	Initial Azimuth Angle
6-DoF	start_psi	Initial Heading Angle
	start_theta	Initial Pitch Angle
	start_phi	Initial Bank Angle
	start_p	Initial Roll Rate
	start_q	Initial Pitch Rate
	start_r	Initial Yaw Rate

2.1 Parameterization

In the context of sequential simulation and therefore changing parameter datasets, it is important to provide a common parameterized dataset interface which is consistent for any derived model complexity. The user-defined record *userPar* has to remain structurally constant in terms of model definition and yet has to remain accessable for the user to change certain parameters during the model redeclaration between two subsequent phases. This is ensured by using the `inner` and `outer` concept of MODELICA as well as extendable records which are based on a mutual baseclass.

For example, in Table 1 a subset of parameters provided by the *userPar* component for the definition of kinematic position, velocity and orientation of the launch vehicle is shown. Depending on the level of detail, the record must contain at least start values for the initial latitude, longitude, altitude, velocity as well as the flight path and azimuth angles for 3-DoF launch vehicle models. For higher levels of detail, the basic parameter dataset can be extended to include additional initial values such as the bodies' orientation angles and roll rates.

2.2 Kinematics & Dynamics

The kinematic state variables chosen for the 3-DoF representation of a launch vehicle system depend on the geocentric position $\boldsymbol{r}_G$ as well as the velocity $\boldsymbol{v}_N$ with respect to the local horizontal coordinate system as shown in Equation (1):

$$\boldsymbol{r}_G = \begin{bmatrix} \text{latitude} \\ \text{longitude} \\ \text{radius} \end{bmatrix}, \quad \boldsymbol{v}_N = \begin{bmatrix} v_{\text{North}} \\ v_{\text{East}} \\ v_{\text{Down}} \end{bmatrix}. \qquad (1)$$

The 3-DoF equations of motion describing the translational launch vehicle dynamics are derived using the sum of external forces. However, the variable mass of the launch vehicle system has to be taken into account by using dedicated variable mass models based on a similar structure as

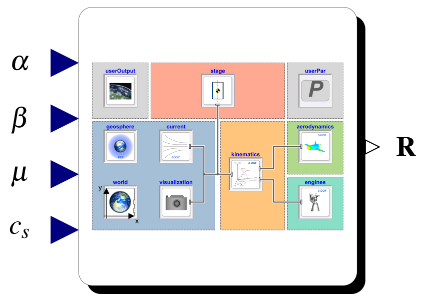


Figure 3. Overview of the Replaceable Main Model provided by the Launch Vehicle Modeling Framework for Sequential Simulation.

the body components in the *Modelica Standard Library*, where the mass m is a state variable defined by the equation $dm = \dot{m}$ and where the mass flow rate dm is calculated using the engine specification data within the *userPar* component.

The external forces consider the gravity G provided by the environment component *world*, the aerodynamic forces A given by the *aerodynamics* component and the thrust force T calculated by the *engines* component. The aerodynamic forces can be determined using aerodynamic coefficients which can be interpolated using multi-dimensional look-up tables. Similar to the aerodynamic forces, the thrust force can be determined using engine specification data or multi-dimensional time-dependent tables.

Since only the translational motion is taken into account, a time-scale separation between translational and rotational dynamics is assumed. Consequently, all angular velocities and accelerations are set to zero. The 3-DoF model is obtained from a constrained 6-DoF point mass model using dedicated kinematic models provided by the DLR Environment Library.

2.3 Model Structure

In this section, a generic model structure which can be used for the sequential simulation of launch vehicle systems is introduced. Therefore, the overall launch vehicle modeling framework as shown in Figure 2 has to be modified for further usage within the sequential simulation.

For instance, to visualize the trajectory of a launch vehicle based on trajectory optimization results or to conduct controllability studies, a set of input and output parameters are required which have to influence the system dynamics. For a 3-DoF launch vehicle model necessary input parameters are generally defined as the angle of attack α, sideslip angle β, bank angle μ, and the engine throttling factor c_s. The output parameters are stored in the result vector **R**. This is schematically shown in Figure 3.

Based on the chosen analysis type of the sequential simulation these input and output parameters along with the replaceable system components and the parameterized dataset have to be adapted to the corresponding analysis requirements.

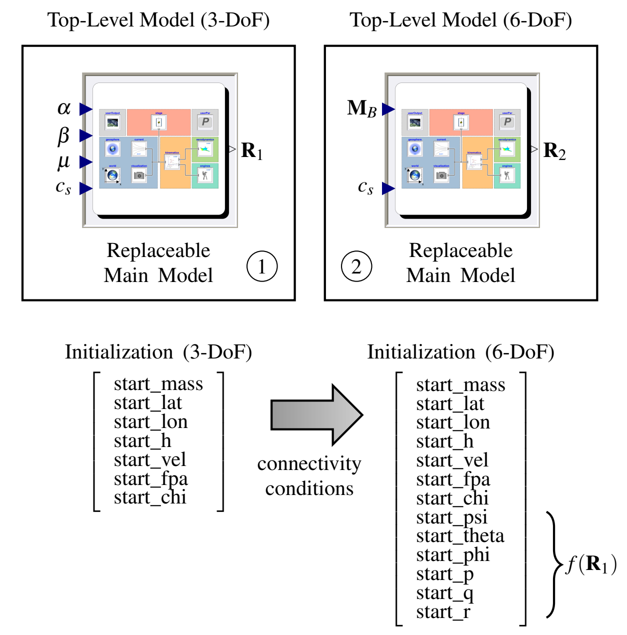


Figure 4. Overview of Multiple Top-Level Simulation Models.

For this purpose, it can be necessary to define a consistent top-level simulation model which is used as an interface during the sequential simulation and where the main launch vehicle modeling framework can be replaced according to the current phase definition and requirements. For example, the simplified top-level simulation model as used in this paper contains the replaceable main model shown in Figure 3 to allow its redeclaration at transition times between phases. For the trajectory visualization, additional multi-dimensional interpolation tables are included in the top-level model to directly provide input parameters for the replaceable main model as obtained by previously performed trajectory optimization. Alternatively, a control structure can be wrapped around the replaceable main model to conduct controllability analyses for specific phases while taking into account uncertainties in the system dynamics.

Another possibility is to define multiple top-level simulation models with consistent interface elements and connectivity conditions to allow the usage of models with different levels of detail. This approach is illustrated in Figure 4 for a sequential simulation containing two different setups for 3-DoF and 6-DoF point mass models. Since the number of states increases from seven for the 3-DoF case to 13 for the 6-DoF case, connectivity conditions as well as additional calculations depending on the output parameters of the previous simulation have to be defined to obtain corresponding initial conditions for the transition between the two top-level models. This approach is similar to the transition conditions as issued for multi-mode DAE systems in (Mattsson et al., 2015).

Since the influence of different levels of detail on the sequential simulation process is out of scope for this paper, this option will not be further investigated here.

3 Sequential Simulation

Within this section, the sequential simulation method based on DYMOLA's built-in command *simulateExtendedModel* will be discussed further. This function simulates a model with modified parameters and provides a subset of results at final simulation time which can be used for subsequent simulations (DYMOLA, 2018). Often, the *simulateExtendedModel* function is used for user-defined parameter studies, in which a certain set of parameter values is propagated into the model using a script-based approach within DYMOLA which is similar to the DYMOLA built-in command *experiment*.

3.1 *simulateExtendedModel* Function

In the DYMOLA documentation, the arguments of the *simulateExtendedModel* function including their default values are defined as shown in Code 1:

Code 1. Arguments of the *simulateExtendedModel* command.

```
simulateExtendedModel(
 problem           = "",
 startTime         = 0.0,
 stopTime          = 1.0,
 numberOfIntervals = 0,
 outputInterval    = 0.0,
 method            = "Dassl",
 tolerance         = 0.0001,
 fixedstepsize     = 0.0,
 resultFile        = "dsres",
 initialNames      = fill("",0),
 initialValues     = fill(0,0),
 finalNames        = fill("",0),
 autoLoad          = true)
```

The *problem* corresponds to the MODELICA path name of the top-level sequential simulation model as shown in Figure 4. Furthermore, the start and stop time of the simulation can be accessed, as well as the number of intervals, the output interval, the integration method, its tolerance and if applicable its fixed stepsize. For each simulation run a unique name for the result file can be chosen. Most importantly, the initial values of chosen parameters defined by their initial names can be set at the start of the simulation and the final values of a user-defined set of parameters can be derived after the simulation run.

This command has been applied for example within the *Hybrid Decomposition* method in combination with DYMOLA-external tools like PYTHON and MATLAB as described in (Mehlhase, 2015) and (Stüber, 2017). For these methods, the online evaluation of the results are indispensable, since the chosen long time simulations depend on event-based information about the end-point of the simulation. Because the sequential simulation models as shown in this paper are constrained in time, the use of DYMOLA-external tools to control the simulation time is not needed. However, the replaceable main models as shown in Figure 3 can be translated into *Functional Mock-up Units* (FMU) and used for multi-phase trajectory optimization in MATLAB as introduced in (Briese et al., 2018).

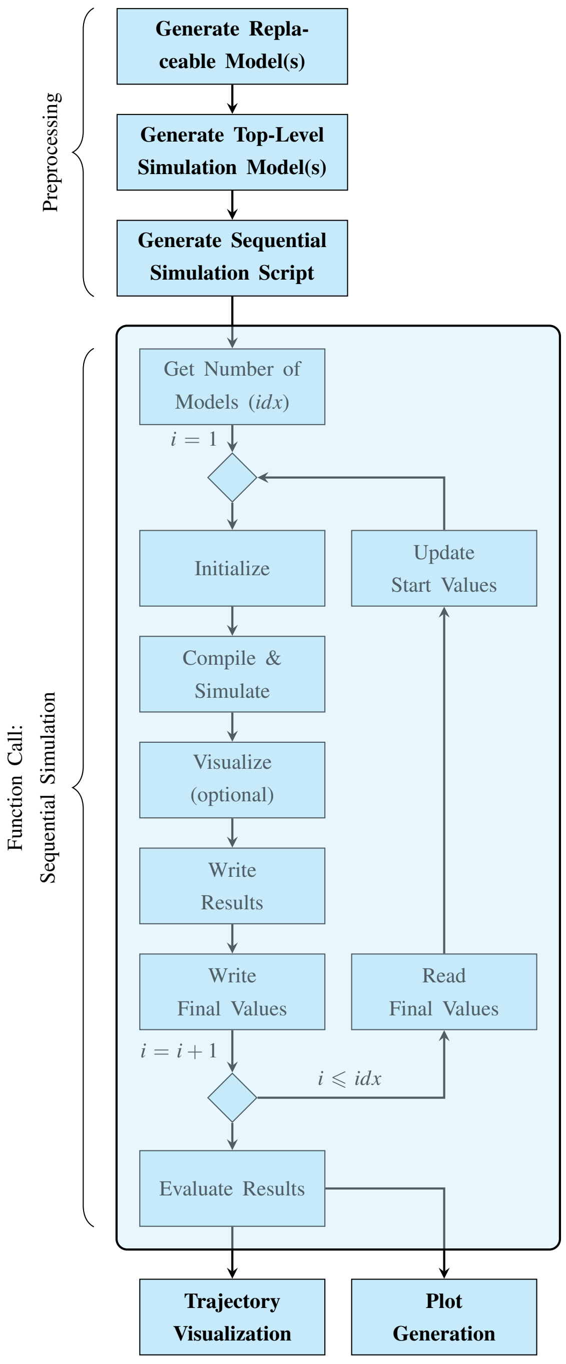


Figure 5. Overview of the Sequential Simulation Method.

3.2 Script-based Sequential Simulation

A basic overview of the sequential simulation method is illustrated in Figure 5. First, several preprocessing steps have to be conducted including the generation of (replaceable) models for each phase, at least one or more top-level simulation models, as well as the corresponding sequential simulation script which is implemented as a function within DYMOLA as shown in Code 2. Since sequential simulation scripts depend on the chosen modeling and simulation purpose, the adaptivity of this script allows a flexible handling of variable model structures and their simulation purpose.

Code 2. Sequential Simulation Script Example.

```
function sequentialScript
 import SI=Modelica.SIunits;
 input SI.Time phaseTimes[:,:];
 input String  stateNames[:];
 input String  modelNames[:];
 input String  exampleName;

protected
 Boolean ok;
 Real out[:];

algorithm
 for i in 1:size(modelNames,1) loop
  (ok,out) := simulateExtendedModel(
   exampleName + "(redeclare " +
     modelName[i] + " test)"
   startTime  = phaseTimes[i,1],
   stopTime   = phaseTimes[i,2],
   resultFile = "phase_" + String(i),
   initialNames  = if i == 1 then
     fill("",0) else stateNames,
   initialValues = if i == 1 then
     fill( 0,0) else out,
   finalNames = stateNames);
  end for;
annotation (__Dymola_interactive=true);
end sequentialScript;
```

Following the preprocessing steps, the sequential simulation script is executed. The current top-level sequential simulation model defined by the DYMOLA path `exampleName` is executed iteratively for each phase using the DYMOLA built-in function *simulateExtendedModel*. The annotation `__Dymola_interactive=true` has to be included into the function call to use the built-in function without errors or warnings.

In this sequential simulation example, the replaceable main models are referenced by their DYMOLA path `modelName`. They can be redeclared within the corresponding top-level sequential simulation model at each phase. Therefore, the top-level model has to be translated and simulated for each phase separately due to possible major changes in the system structure of the replaceable main models. This is for example the case, if for each phase different datasets have to be obtained from external sources or if the number of states changes.

The arguments of the DYMOLA command *simulateExtendedModel* allow for the initialization of certain parameters within the simulation model and to store chosen final values as a function output in the vector `out`. Within the sequential simulation method, these arguments are only used for the initialization of state variables depicted by the `stateNames` parameter since any other changes within the models are defined using either the parameter setup in the record `userPar` or direct parameter propagation. In the first iteration, the state variables are initialized using the start values stored within the record *user-*

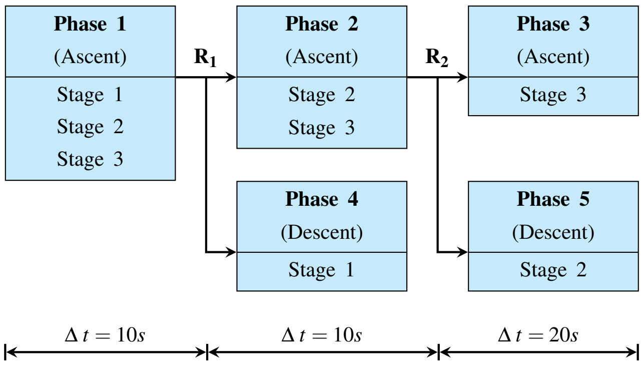
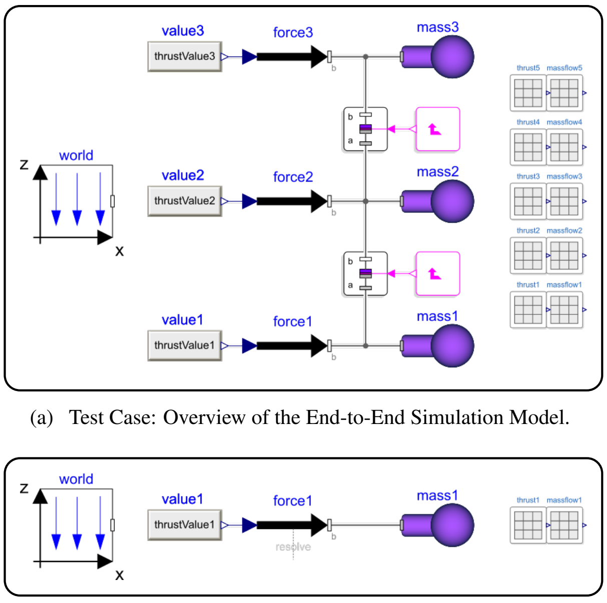


Figure 6. Test Case: Overview of the Phases.



(a) Test Case: Overview of the End-to-End Simulation Model.

(b) Test Case: Overview of the Sequential Simulation Model.

Figure 7. Test Case: Overview of the Generic Model Structure.

Par. Therefore, the function parameters `initialNames` and `initialValues` are declared as empty vectors. Additionally, the simulation time limits given by `startTime` and `stopTime` depend on user-defined inputs or alternatively on the final time of the previous model.

For each phase, a result file containing only the state variables, their derivatives as well as input and output variables is saved. Depending on the application requirements, the result files can be further reduced by deactivating certain outputs for example if the sequential simulation method has to be used in an optimization. At the end of the sequential simulation script the result files are evaluated for further post-processing steps. For example, the post-processing step can include an overall trajectory visualization based on the final result files using the visualization components provided by the DLR Visualization Library (Bellmann, 2009) and SimVis as shown in Figure 11 in Section 4.

4 Results

In this section, the sequential simulation method is applied for two different launch vehicle concepts. First, a generic test case is introduced to demonstrate the capabilities of the sequential simulation method while using a rather simplified multibody model. Second, the delta-winged reusable launch vehicle concept AURORA is modeled and visualized while using the sequential simulation method.

4.1 Test Case

To demonstrate the capabilities and advantages of sequential simulation, a simplified yet generic 6-DoF multibody model with three stages, two separations and therefore five dedicated phases has been analyzed as depicted in Figure 6. For better comparability, two dedicated models for the end-to-end as well as the sequential simulation concept have been generated as shown in Figure 7.

The end-to-end simulation model contains three stages represented by variable point mass models. Each stage is connected to the next stage with separation components. As long as the boolean signal is defined as `false` the stages remain connected and they are separated if otherwise (see (Acquatella B. and Reiner, 2014)).

The mass flow rate and thrust magnitude for each stage are obtained from MATLAB files using multi-dimensional time-dependent interpolation tables to include a realistic modeling challenge since import and interpolation of external engine or aerodynamic datasets have to be performed very often in design studies. For this purpose, interpolation tables for each phase have to be included into the simulation model. The thrust is applied only in vertical z-direction such that a vertical ascent of all stages is required since only uniform gravity in z-direction and no side forces are applied on the point mass models. Additionally, the thrust and mass flow rate are set to zero using state dependent conditional statements if the overall mass of the components reaches a certain threshold.

The sequential simulation model on the other hand only provides one variable mass point to represent the stages. Consequently, the individual mass contribution of each stage is summed up into an overall mass representation of the body. This is generally the case if the connection between each rigid stage is assumed as an ideal and not an energy-consuming connection. Additionally, only the interpolation data for the current phase is needed and can be referenced directly by changing the table name to obtain correct results for the corresponding stages.

The function introduced in Code 2 is a simplified example for a generic sequential simulation. Any other variation of the sequential simulation script can be implemented according to the use case. For instance, the script has to be adapted to new requirements if splitted trajectories are taken into account as presented in Figure 6. In this case, the first phase is simulated providing the output vector $\mathbf{R_1}$ for the final states. This output vector is then used to initialize the states for the second and fourth phase while using fixed start values for the mass as defined by

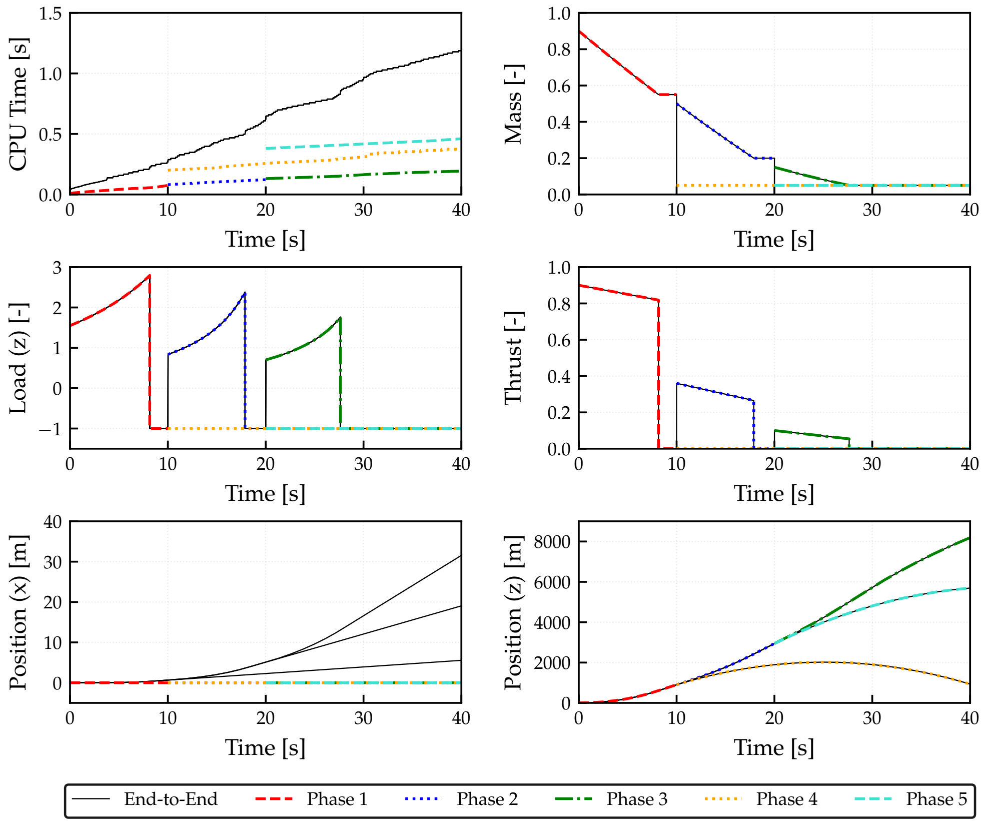


Figure 8. Test Case: Results obtained for the End-to-End as well as the Sequential Simulation Concept.

Table 2. Test Case: Comparison of Simulation Parameters.

	End-to-End	*Sequential*
Integration method	dassl	dassl
Integration Tolerance	$1e-7$	$1e-7$
Number of States	39	13
Number of State Events	3	3 (total)
Number of Time Events	5	3 (total)
Number of Equations	3791	1134

the corresponding stage structure. The same approach is considered for the third and fifth phase using the second output vector $\mathbf{R}_2$.

In Table 2 an overview of general simulation parameters for both modeling and simulation concepts is given. For better comparability both simulation setups use the same integration method and tolerances as well as similar number of intervals. In the sequential simulation case, the simulation parameters are shown for one phase. Only the number of state and time events are given as a sum over all phases for the sequential simulation. The number of state events cannot be reduced since they are used to conditionally set the mass flow rate to zero if a certain threshold for the propellant mass in the variable point mass models is reached. On the other hand, the time events can be reduced. Three time events result from the time-dependent table interpolation while the other two are needed to define the point of time where separation occurs. Another advantage of the sequential simulation concept in this context is that it only contains linear systems of equations, while the end-to-end simulation model has to initialize and solve two nonliner systems of equations due to the separation components.

The number of equations and states for each simulation phase can be significantly reduced by only using one single point mass model and by removing the separation components as well as additional table interpolation components. Obviously, this is only true if one specific phase model is considered. But since the *world* model has to be considered separately in each phase of the sequential simulation and since the end-to-end simulation model uses only three variable point mass models compared to five for the overall sequential simulation process, the number of equations is higher and can therefore also result in higher compilation times. Although the compilation time of variable structure models plays a major role regarding comparability aspects, the additional compilation time is not further considered since the focus of the paper is set on the reduction of the model complexity.

However, the overall CPU time can be reduced significantly as can be seen in Figure 8 summed up for all sequentially simulated phases. The simulation time is especially sensitive towards the chosen number of integrals for each phase. While this parameter can be only defined once for the end-to-end simulation, the number of intervals for each phase can be chosen individually according to the simulation needs. For instance, if the third phase is not relevant for the overall simulation purpose, this parameter could be decreased in order to reduce the simulation time which would not be possible for an end-to-end simulation concept.

As shown in Figure 8, the results for both concepts are the same regarding loads, thrust, mass flow rates as well as the position in z-direction of each body component except for the translational drift in the horizontal x-direction after each separation while using the end-to-end simulation model. Since all forces are applied only vertically in the bodies' z-direction according to a flat *world* assumption, this drift error should be ideally zero as can be seen by the results of the sequential simulation models. The resulting drift of the multibody parts after separation is caused by the acceleration residual at separation time, which in turn is caused by the *Baumgarte stabilization* solution during joint body motion whose accuracy is sensitive to computational erros as mentioned in (Acquatella B. and Reiner, 2014).

4.2 Use Case

In this section, the results obtained by the sequential simulation method as applied for the *horizontal takeoff and horizontal landing* (HTHL) delta-winged launch vehicle AURORA are presented.

A generic trajectory for a winged reusable launch vehicle configuration is illustrated in Figure 9. Here, the ascent phase can be divided into multiple phases which can include an horizontal takeoff, a powered ascent including the gravity turn as well as MECO after which an intermediate ballistic flight phase can follow. After stage separation the upper stage continues with the ascent into the desired orbit, including several phases which can be constrained by fairing and payload separation. For descent, the flyback vehicle and its phase definitions are mainly dependent on the descent configuration and the horizontal or vertical landing mode.

This reusable launch vehicle concept as described by (Kopp et al., 2017) consists of seven dedicated phases as shown in Table 3. The definition of these phases depends highly on the mission and has to be considered before the trajectory optimization. The results obtained with the Trajectory Optimization Package *trajOpt* by (Schnepper, 2014) and the multi-objective and multi-phase optimization tool MOPS as described in (Joos, 2016) will be used in the following.

The top-level simulation model as shown in Figure 4 remains the same for each phase and only the replaceable main models as described in Section 2.3 are redeclared within the sequential simulation script using a similar script structure as shown in Code 2. The fixed start and final time values are provided by the results of the trajectory optimization. The state definition of all models corresponds to the state vector given in Equation (1) including the varying overall mass of the launch vehicle system.

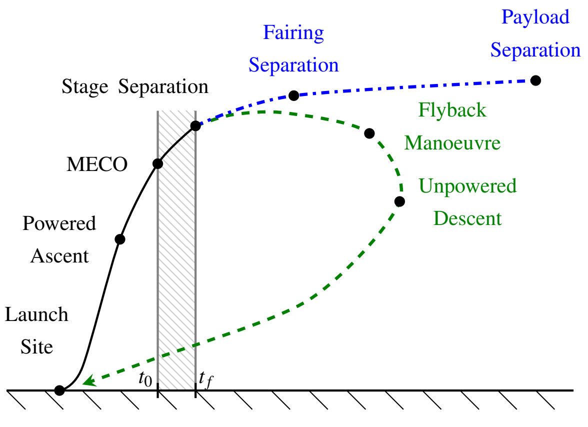


Figure 9. Schematic Trajectory of a Winged Launch Vehicle.

Table 3. Use Cases: Overview of the Phases (AURORA).

Phases	*Stages*	*Description*
Phase 1	US+MS	Horizontal liftoff
Phase 2	US+MS	Ascent phase (rocket engines)
Phase 3	US+MS	Ballistic phase & separation
Phase 4	US	Ascent of the upper stage
Phase 5	MS	Return maneuvre 1
Phase 6	MS	Return maneuvre 2
Phase 7	MS	Return to the launch site

Furthermore, the initialization of the models can be realized either using the output vectors of previously simulated phases or the start values of the trajectory optimization results for each phase. Consequently, the iterative call of the *simulateExtendedModel* function in Code 2 is divided into three parts:

1. **Sequential Simulation of Phases 1 to 3**:
   Phase 1 is initialized by itself using default values defined by parameters in the *userPar* component. Phase 2 and 3 are initialized depending on the final values of the corresponding previous phases. The final values of Phase 3 at its end-point are stored in a separate output vector `out_sep`.

2. **Simulation of Phase 4**:
   This phase is initialized using only the position- and velocity-related final values stored within the `out_sep` vector. The state parameter referencing the mass of the system is overwritten by the overall mass of the upper stage (US).

3. **Sequential Simulation of Phases 5 to 7**:
   Phase 5 is initialized using the position- and velocity-related final values stored within the `out_sep` vector. The parameter referencing the mass state of the system is overwritten by the overall mass of the main stage (MS). The last two phases are initialized depending on the final values of the corresponding previous phases.

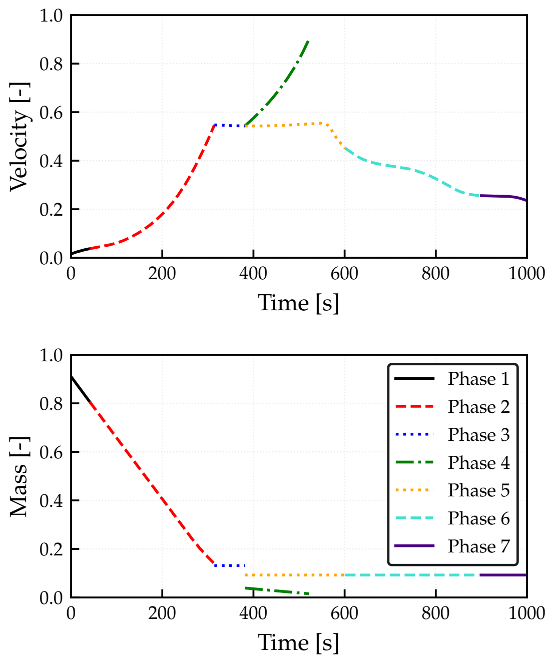


Figure 10. Use Case: Results obtained for each Phase (AURORA).

The main challenges in the modeling of this reusable launch vehicle concept are related to changing engine configurations as well as aerodynamic characteristics for different flight conditions. For this purpose, at least two different aerodynamic datasets for each aerodynamic coefficient have to be considered which can be either used for the ascent or the descent phase depending on the vehicle's current Mach number and angle of attack. Using the replaceable main models, simply one aerodynamic dataset has to be incluced into the current simulation model, which can reduce the overhead of parameters used within the compiled top-level simulation model.

Since a change of the number of states is not required within this simulation, the overall system configuration can remain the same. Therefore, the same replaceable main model setup can be used by reconfiguring the parameter initialization and importing different datasets for each phase. This way, the same number of states and equations is generated which reduces the model complexity as well as the overhead of unnecessary calculations and also leads to a consistent model structure througout the sequential simulation.

In addition to the results presented in (Briese et al., 2018), the normalized velocity and the overall mass of the vehicle during the sequential simulation are shown in Figure 10. The overall mass within each phase is varying continuously based on the mass flow rate obtained by the vehicle's engine specification. Due to the separation after the third phase at approximately $390s$ steps occur in the mass formulation resulting from the reinitialization of the masses of the upper stage (US) in Phase 4 and the winged flyback stage (MS) in Phase 5. While these step commands do not effect the integration within each phase, these kind of steps can lead to numerical errors within end-to-end simulations and would have to be smoothed using approximation functions by increasing the model complexity. Furthermore, as shown by the velocity results, the transition between each phase is performed smoothly by initializing all kinematic state variables consistently.

Finally, the result of the subsequent trajectory visualization performed with the DLR Visualization Library as a post-processing step is shown in Figure 11. In this case, the ascent of the combined stages (Phase 1 to 3), the ascent of the upper stage (Phase 4) as well as the descent of the winged main stage (Phase 5 to 7) are visualized separately using the corresponding post-processed result files of the sequential simulation.

5 Conclusion

In this paper, the script-based sequential simulation method based on DYMOLA's built-in command *simulateExtendedModel* has been introduced.

The purpose was a simplification of end-to-end simulations containing time- and state-dependent conditional statements and separation models to handle the transition within variable structure launch vehicle models. For this purpose, the replaceable launch vehicle models within the launch vehicle modeling framework have been introduced and the implementation of these models within top-level simulation models to be used by the sequential simulation has been shown. The significance of a common and consistently defined parameter set has been underlined.

Additionally, the sequential simulation method has been further investigated and subsequently applied on a winged reusable launch vehicle configuration. The results show that the transition conditions between each phase can be determined using the initialization arguments within the *simulateExtendedModel* function. Furthermore, time- and state-dependent conditional statements as well as the usage of separation models can be avoided. Also, the number of equations remains the same for each phase and can be significantly lower than the overall launch vehicle model for end-to-end simulations.

As an example regarding the application options for the sequential simulation method, the subsequent trajectory visualization as a post-processing step in DYMOLA was presented for the AURORA launch vehicle configuration.

For future research, the *Hybrid Decomposition* method could be extended by using DYMOLA-internal methods. Also, the simulation arguments could be designed flexibly depending on the simulation characteristics to dynamically adapt to the computational requirements of each phase.

Acknowledgements

This study was performed within the DLR internal project AKIRA for the comparison of key technologies for future reusable launch vehicles.

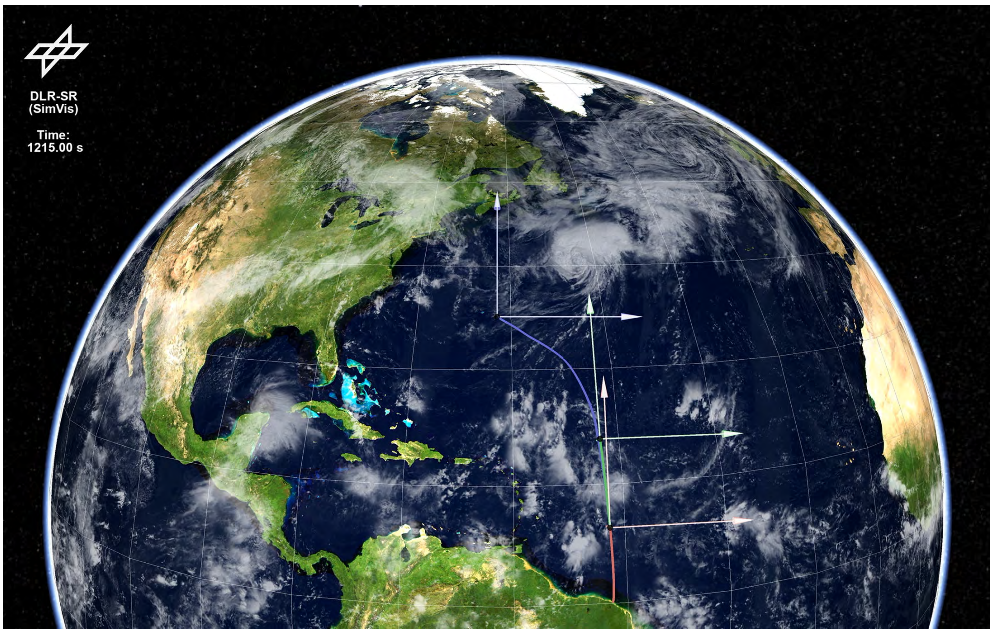


Figure 11. Trajectory Visualization of the Reusable Launch Vehicle AURORA.

References

P. Acquatella B. Launch Vehicle Multibody Dynamics Modeling Framework for Preliminary Design Studies. In *6th International Conference on Astrodynamics Tools and Techniques, ICATT*, 2016.

P. Acquatella B. and M. J. Reiner. Modelica Stage Separation Dynamics Modeling for End-to-End Launch Vehicle Trajectory Simulations. In *Proceedings of the 10th International Modelica Conference*, 2014.

T. Bellmann. Interactive Simulations and advanced Visualization with Modelica. In *Proceedings of the 7th International Modelica Conference*, 2009.

L. E. Briese, A. Klöckner, and M. Reiner. The DLR Environment Library for Multi-Disciplinary Aerospace Applications. In *Proceedings of the 12th International Modelica Conference*, 2017.

L. E. Briese, K. Schnepper, and P. Acquatella B. Advanced Modeling and Trajectory Optimization Framework for Reusable Launch Vehicles. In *Proceedings of the IEEE Aerospace Conference (in press)*, 2018.

DYMOLA. *Version 2018*. Vélizy-Villacoublay, France, Dassault Systèmes, 2018.

H. Elmqvist, S. E. Mattsson, and M. Otter. Modelica extensions for Multi-Mode DAE Systems. In *Proceedings of the 10th International Modelica Conference*, 2014.

H.-D. Joos. MOPS - Multi-Objective Parameter Synthesis. Technical Report DLR-IB-SR-OP-2016-128, DLR German Aerospace Center, 2016.

A. Kopp, M. Sippel, S. Stappert, N. Darkow, J. Gerstmann, S. Krause, D. Stefaniak, M. Beerhorst, T. Thiele, A. Gülhan, R. Kronen, K. Schnepper, L. E. Briese, and J. Riccius. Forschung an Systemen und Technologien für wiederverwendbare Raumtransportsysteme im DLR-Projekt AKIRA. In *Deutscher Luft- und Raumfahrtkongress*, 2017.

S. E. Mattsson, M. Otter, and H. Elmqvist. Multi-Mode DAE Systems with Varying Index. In *Proceedings of the 11th International Modelica Conference*, 2015.

A. Mehlhase. *Konzepte für die Modellierung und Simulation strukturvariabler Modelle*. PhD thesis, Technical University Berlin, Germany, 2015.

A. Mehlhase, D. G. Esperon, J. Bergmann, and M. Merkle. An example of beneficial use of variable-structure modeling to enhance an existing rocket model. In *Proceedings of the 10th International Modelica Conference*, 2014.

Modelica Association. *Modelica - A Unified Object-Oriented Language for Physical Systems Modeling, Language Specification Version 3.3. Revision 1*, 2014.

M. J. Reiner and J. Bals. Nonlinear Inverse Models for the Control of Satellites with Flexible Structures. In *Proceedings of the 10th International Modelica Conference*, 2014.

K. Schnepper. Trajektorienoptimierung in MOPS - Das Paket trajOpt Version 1.0. Technical report, DLR German Aerospace Center, 2014.

M. Stüber. Simulating a Variable-structure Model of an Electric Vehicle for Battery Life Estimation Using Modelica/Dymola and Python. In *Proceedings of the 12th International Modelica Conference*, 2017.

D. Zimmer. *Equation-Based Modeling of Variable-Structure Systems*. PhD thesis, ETH Zurich, Switzerland, 2010.