UNIFIER19 Flight Dynamics Simulator Model Documentation

J. Soikkeli, D. Matko, T. Koopman

20^{th} July 2022

Contents

List of Figures

List of Tables

1 Introduction

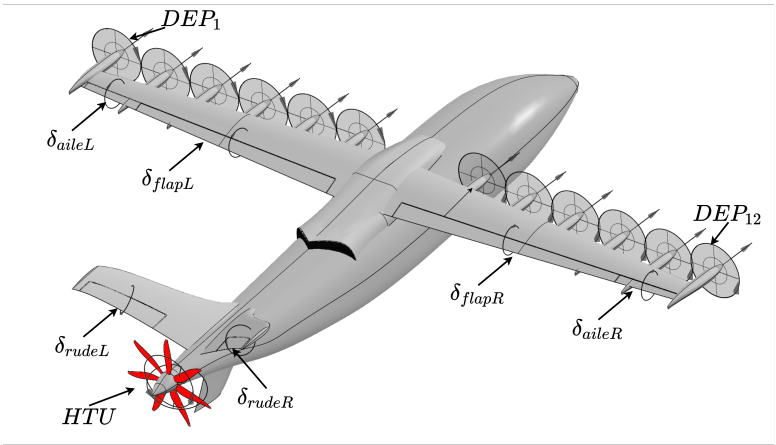


Figure 1: Effectors of the UNIFIER19 C7A-HARW

UNIFIER19 is a partnership project to evaluate a conceptual hydrogen powered 19-seat miniliner. As a part of the project flight dynamics simulator and a flight control laws were developed. More information about the vehicle and project can be found on the project websites[1]. Figure 1 illustrates the effectors of the UNIFIER19 C7A-HARW. It can be seen that the vehicle has six independent control surfaces. The vehicle contains also 12 fixed pitch foldable distributed electric propulsors (DEP) mounted on the leading edge. Finally, the vehicle contains one variable pitch main propulsors, also called a horizontal thrust unit (HTU), mounted at the aft of the fuselage. Tasking a pilot to control all of the effectors is impractical, hence for a vehicle like C7A-HARW, an advanced fly-by-wire system is needed with appropriate control laws to reduce the pilot workload.

As a part of the UNIFIER19 project, we developed flight control laws based on nonlinear dynamic inversion. Paper is to be published on the findings.

With this model, we invite you to design your control laws! We hope this simulator will serve as an easy go-to simulator for DEP vehicles! Looking forward to your results!

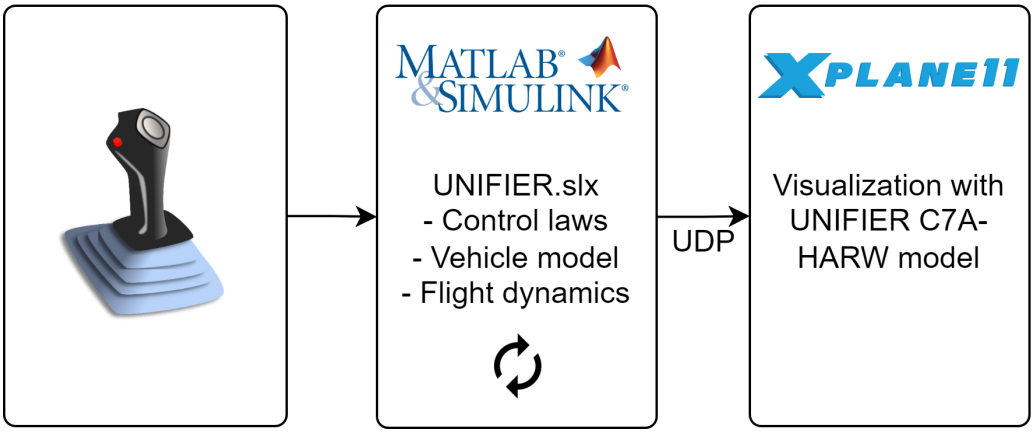


Figure 2: UNIFIER19 C7A-HARW flight dynamics simulator

The project has received funding from the CleanSky2 Joint Undertaking (JU) under grant agreement NO 864901. The JU receives support from the European Union's Horizon 2020 research and innovation programme and the Clean Sky 2 JU members other than the Union.

[1] `https://www.unifier19.eu/?page_id=245`

2 Aerodynamic Data Collection

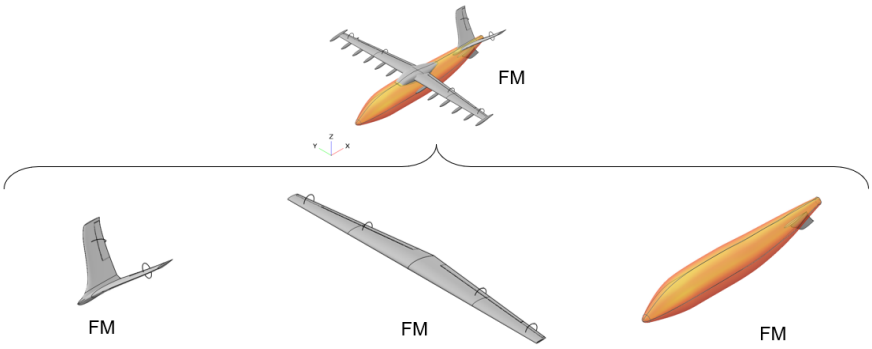


Figure 3: UNIFIER19 C7A-HARW component split for the aerodynamic data collection

In this section the collection of the aerodynamic data is discussed. The aerodynamic data is collected with FlightStream; A novel panel method capable of evaluating aero-propulsive interactions and containing boundary layer models.

The root idea of the aerodynamic data collection of the C7A-HARW is to split the aircraft into components and each component into segments. For each segment there exist a multi-dimensional look-up table, which is used to evaluate the forces generated by the segment itself. The forces and moments generated by each segment are summed to derive the total forces and moments acting in the vehicle point mass. The advantage of the segmentation, compared to evaluating aerodynamic data for an entire vehicle, is that each segment can have unique inputs based on the operating conditions of the very segment. The aerodynamic characteristics of DEP vehicle, like C7A-HARW, are highly coupled. This means that i.e. DEP propulsion, wing, and flaps are all interacting with each other. Hence, to keep the dependencies manageable, the component and segment split is seen as a good solution. If the aerodynamic data would be gathered for the entire (no component split) vehicle, the computational effort would be enormous to capture all the dependence. In order of weeks to months with a regular desktop PC. Hence, the component split is essentially an effort to reduce computational time.

The aerodynamic interactions of the component split can be split into intra-component interactions (i.e. DEP interaction with wing) and extra-component (i.e. wing interaction with V-tail). One criticism of the component split approach is that the extra-component interactions are not accounted for as the aerodynamic data is collected separately for each component. It is argued that the interaction between wing and fuselage can be safely neglected. The same holds for the V-tail and fuselage interaction. However, the wing and the V-tail interaction cannot be neglected. The main impact of this interaction is the down-wash at the V-tail caused by the wing. In the implemented methodology this interaction is captured by recording the wing down-wash at the location of the V-tail.

2.1 V-tail

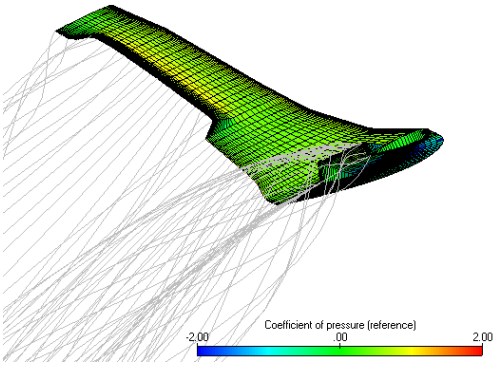


Figure 4: V-tail in FlightStream software

The forces generated by the V-tail are exported as surface force quivers ($\mathbf{F}_{VTAIL}$). In other words, the forces of each panel of the mesh are saved. Exporting the surface force quiver allows the segmentation to be done in post-processing. The independent variables for the V-tail are the angle of attack, sideslip, and rudevon deflection. The data collection is expressed as:

$$\mathbf{F}_{VTAIL}(\alpha, \beta, \delta_{rude})$$

$$\begin{aligned} \alpha &= [-15, -10, -5, -2.5, 0, 2.5, 5, 7.5, 10, 12.5, 15, 17.5, 20] \\ \beta &= [-15, -10, -5, -2.5, 0, 2.5, 5, 10, 15] \\ \delta_{rude} &= [-30, -25, -20, -15, -10, -5, 0, 5, 10, 15] \\ &= 13 \times 9 \times 10 = 1170 \text{ points} \end{aligned} \tag{1}$$

Each aerodynamic solution takes around 45 seconds which results in around 14 hours of total computation time. Note that the rudevon deflection is symmetrical for both of the rudevons. The setting used in FlightStream can be seen in table 1.

Table 1: The FlightStream settings for V-tail data collection

FlightStream version:	#4252022
Viscous-coupling:	Enabled
Drag module:	Reynolds Averaged
S:	29.98 m^2
c:	1.436 m
b:	20.11 m
V:	75 m/s

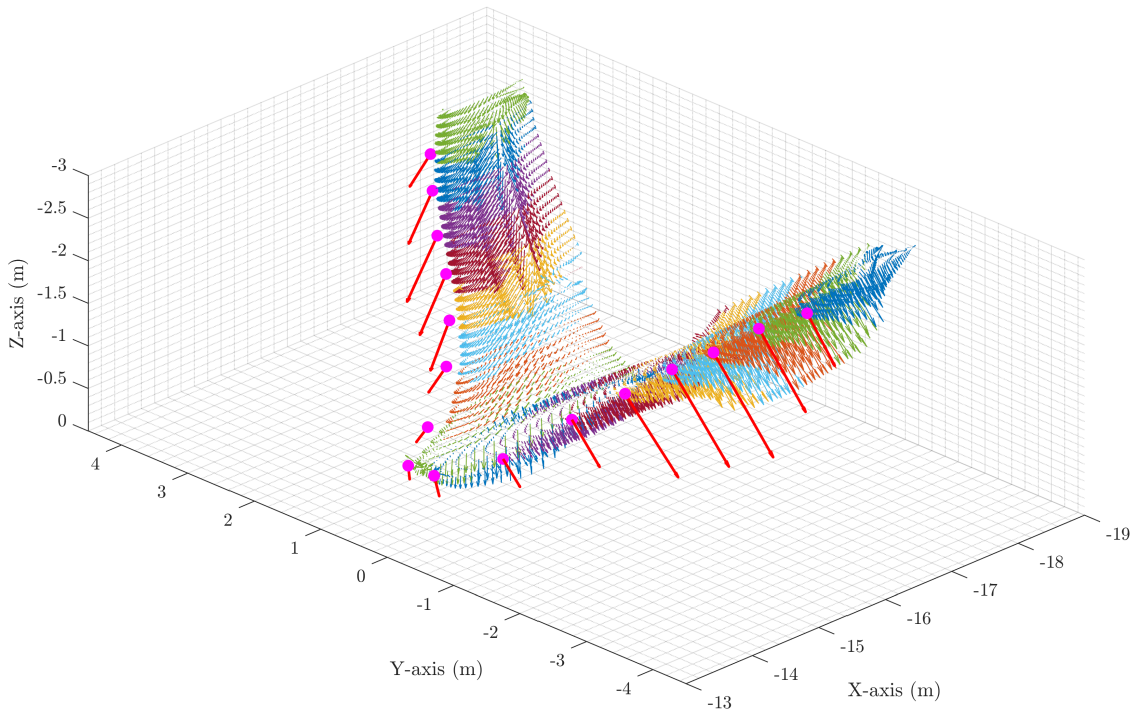


Figure 5: V-tail segmentation for a case with $\alpha = 2°, \beta = 0° \delta_{rude} = -30°$

In post-processing, the surface force quivers are split into segments. The V-tail is split into 16 segments (Figure 5). To generate the segment data, the individual panel forces (for each segment) are summed around a reference point which is selected to be the leading edge of each segment. The summation is visualized with the red resultant force vectors in the figure. Also, the moment around the segment reference point generated by the panel forces is computed and summed. Note that the selection of reference point location does not actually matter, as any selected point will result in correct force and moment representation. The influence of the wing wake on the V-tail is captured in the same locations as the reference points. Finally, force and moment tables are stored for each segment and the resulting aerodynamic data for the V-tail is expressed as:

$$\begin{gathered} C_{x,y,z}(\alpha, \beta, \delta_{rude}, xyz_{ref}^{vtail}) \\ CM_{x,y,z}(\alpha, \beta, \delta_{rude}, xyz_{ref}^{vtail}) \\ xyz_{ref}^{vtail} \end{gathered} \tag{2}$$

2.2 Wing

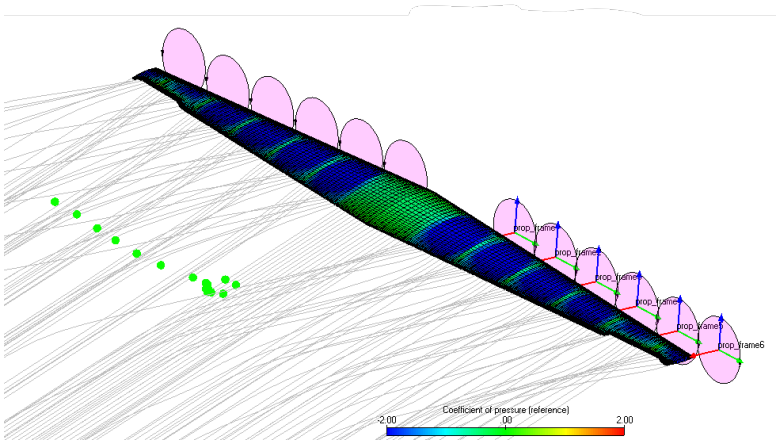


Figure 6: Wing in FlightStream software

Similarly to V-tail, the wing forces are exported as force quivers. In addition to the force quivers, the induced velocity is velocity is recorded at the locations of the V-tail segment reference points. The induced velocity data is captured with the help of FlgithStream probe points and it is used to determine the impact of the wing on the V-tail. The wing data is collected for independent variables: δ, α, V, and J and is expressed as:

$$
\begin{aligned}
&\mathbf{F}_{wing}(\delta, \alpha, V, J) \\
\delta_{flap} &= [0, 5, 10, 15, 25] \\
\delta_{aile} &= [-25, -15, -10, -5, 0, 5, 10, 15, 25] \\
\alpha &= [-15, -10, -5, -2.5, 0, 2.5, 5, 7.5, 9, 10.5, 12, 13.5, 15, 17.5, 20] \\
V &= [30, 50, 90] \\
J &= [0.6, 0.7, 0.8, 1.0, 1.4, 1.8, \infty] \\
&= 13 \times 15 \times 3 \times 7 = 4095 \text{ points}
\end{aligned}
\tag{3}
$$

It can be seen that the wing data is a 5D look-up table. The deflection cases are split such that first the flap is deflected with no aileron deflection, which after the aileron is deflected with no flap deflection. Note that both the flap and aileron cases can use the same zero deflection case, hence, there are only 13 deflection cases (instead of 14). The wing is segmented into 7 segments, and the data is further split into data packets labeled *dp_wing_ROOT* and *dp_wing_TIP*. The root data contains the flap deflections and the tip data contains the aileron deflection. Note that only segments 1:5 are stored in the root data as these are the segments impacted by the flap deflections. Similarly, only segments 6:7 are stored in the tip data as these are the segments impacted by the aileron deflection. The wing segmentation can be seen from Figure 7. Note that one of the DEP propellers is split into two segments because of the control surface segmentation. Hence the influence of this propeller is contained in both of these segments.

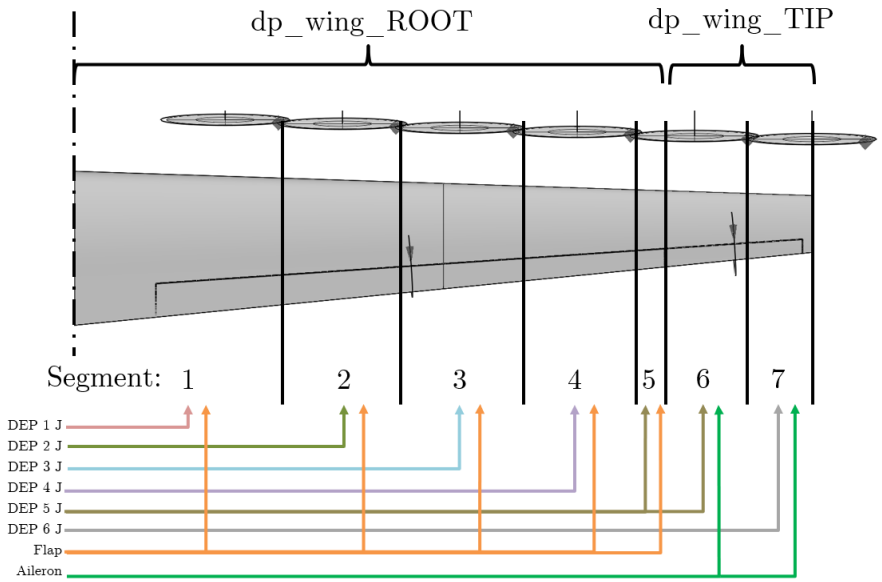


Figure 7: Wing segmentation

The velocity dependency of the data is very small. The only source of velocity dependency is propeller RPM. For an isolated propeller, the advance ratio is fully independent of velocity (neglecting Reynolds effects and compressibility for now). However, when wing is present, propellers with same advance ratio but different airspeed will have the same propeller performance, but the propellers will rotate at different RPM. As an example case with $V = 50\ m/s$ and $V = 90\ m/s$. If in both cases the propeller operates at $J = 0.7$ the propeller will generate same performance but the RPMs are differ: 2678 RPM ($V = 50\ m/s$) and 4821 RPM ($V = 90\ m/s$). The impact of the RPM is on the swirl component. In practice, the impact of this variation of swirl on the wing lift is relatively small. Hence, the velocity dependency could be almost removed from the data. However, it is decided to be kept. The interested reader is referred to [1] for future discussion on the velocity dependency of DEP propellers.

Visualization of the segmentation can be seen from Figure 8, which is a flap deflection case, and hence there is no computation of the resultant force for segments 5:6. Note that even though in the visualization only the resultant forces are shown, there exist also similarly resultant moments. The induced velocity on the vertical tail segment reference points can be seen in the figure. The arrows show the change in flow from the free-stream conditions (in other words, if there is no wing present, there are no arrows). The sign of the induced components is reversed for visualization purposes. It can be seen that in this particular case the propeller slipstream is impacting the V-tail and hence some of the probes are encountering significant induced velocity. The rest of the probes are seen to be encountering a small (compared to the propeller slipstream) down-wash from the wing.

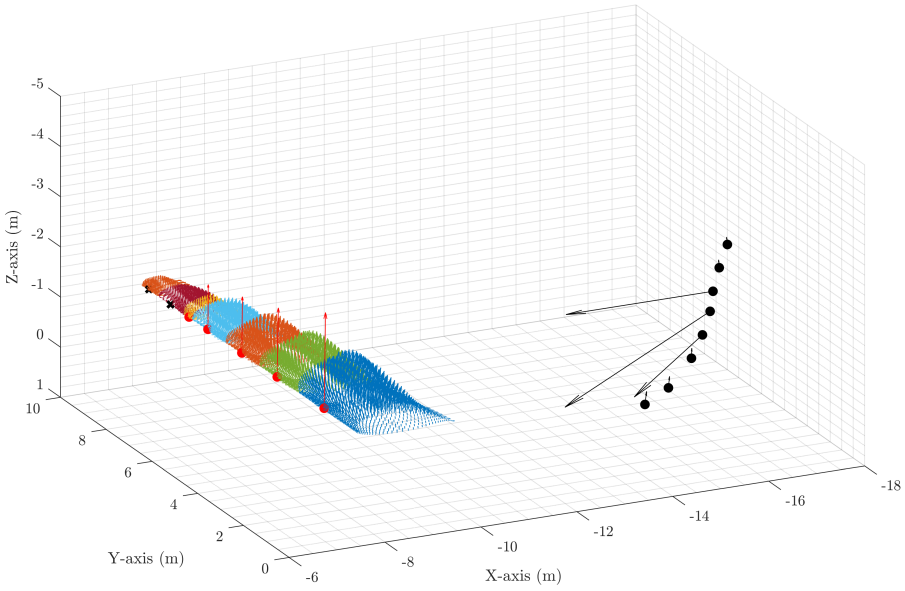


Figure 8: Wing segmentation and induced velocity probes at $\delta_{flap} = 10°$, $\alpha = 0$, $V = 50$, $J = 0.6$. Note that this is a flap deflection case, hence, no computation of resulting force for segments 5-6.

After the segmentation described above the aerodynamic data can be expressed as:

$$\begin{gathered} C^{ROOT}_{x,y,z}(\delta_{flap}, \alpha, V, J, xyz^{wing}_{ref}(1:5)) \\ C^{TIP}_{x,y,z}(\delta_{aile}, \alpha, V, J, xyz^{wing}_{ref}(6:7)) \\ CM^{ROOT}_{x,y,z}(\delta_{flap}, \alpha, V, J, xyz^{wing}_{ref}(1:5)) \\ CM^{TIP}_{x,y,z}(\delta_{aile}, \alpha, V, J, xyz^{wing}_{ref}(6:7)) \\ xyz^{wing}_{ref} \end{gathered} \tag{4}$$

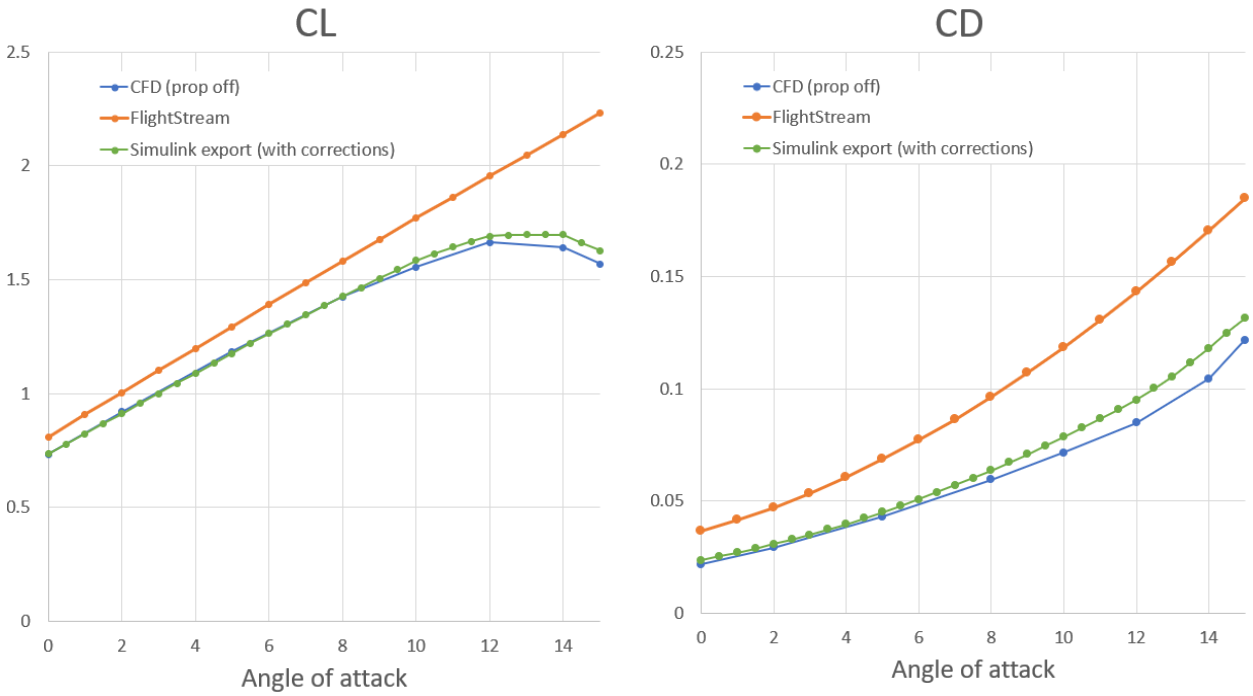
$$\begin{gathered} \alpha_{induced}(\delta_{flap}, \alpha, V, J, xyz^{vtail}_{ref}(1:14)) \\ V_{induced}(\delta_{flap}, \alpha, V, J, xyz^{vtail}_{ref}(1:14)) \end{gathered}$$

There exists several limitations to the aerodynamic analysis of the wing. Firstly, in order to reduce the computational time, the wing was mirrored around the symmetry plane. This reduced the mesh to half and approximately halves the computational time. However, this means that no sideslip dependency can be collected. This means that the wing's anhedral angle is not affecting the aerodynamic model. This limitation is planned to be removed in the future by adding a sideslip dimension to the data. This, however, will require a significant upgrade in computational power.

The second limitation is related to the segmentation assumption. It is assumed that there exist no segment-to-segment interactions, which allows the interpolation of the segments independently of each other. This is not true in reality as the segments do influence also neighboring segments, especially in the case of DEP propulsors. Note that the data is gathered with all DEPs operated in unity, and hence the error of this segmentation assumption is more prevalent when there exist large advance ratio differences between each segment. Another example is stall, the stall can spread from one segment to another, which will not happen if the segments are assumed to be independent.

2.3 Wing Data Post-Processing

As the wing data is seen as vitally important to the overall performance of the vehicle, the aerodynamic data collected with FlightStream is corrected to match the CFD runs performed at TUDelft for the same wing. The correction to the data is applied in to both lift and drag coefficient. Especially, it is seen that the stall of the wing is not correctly captured by FlightStream. Therefore FlightStream is run with the viscous-coupled solver turned off (as a result FlightStream results are linear with no stall) and the non-linearity of lift and stall is applied in the forms of correction factors based on TUDelft CFD data.



(a) The lift coefficient of the non-blown clean wing

(b) The drag coefficient of the non-blown clean wing

Figure 9: The aerodynamic correction data

Figure 9 shows the total lift and drag coefficient of the main wing in non-blown conditions with no flap deflection. The CFD refers to the high fidelity CFD data from TUDelft. It can be seen that the raw FlightStream is data is fully non-viscous. The correction factor is defined as a function of the angle of attack and applied to all of the segment for all of the dependent variables. The Simulink export shows the final result of the data after corrections. It was seen that applying the same correction for the blown conditions resulted in good match with CFD. Due to lack in available CFD data no evaluation was performed for the flap deflection cases. Consequently it is possible that the flap performance is somewhat mischaracterized. In ideal case there would be separate correction factor for each dependent dimensions, however, due to limitation in time the focus was kept on angle of attack and advance ratio dimensions. This is seen justifiable, as the main interest of DEP vehicle is in the blown effect.

2.4 Fuselage and Ventral Fin

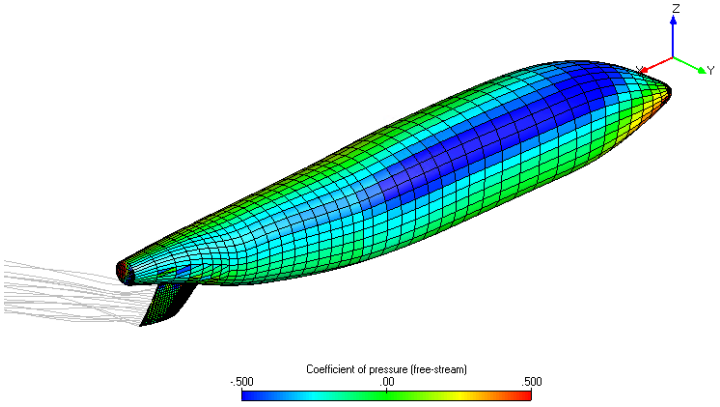


Figure 10: Fuselage and ventral fin data collection.

In the case of the fuselage and ventral fin, there is no need for segmentation because there are no other dependencies than the angle of attack and sideslip. Also, there are no intra-component dependencies that need to be captured with segmentation. Hence the results are exported directly as force and moment coefficients from FlightStream. Note that a significant simplification is made by assuming that the wing will not interact with the fuselage and vice versa. This assumption is seen as justifiable as the purpose of the simulator is to study flight dynamics phenomena. The fuselage and ventral fin data is represented as:

$$\begin{aligned} C_{D,S,L}(\alpha,\beta) \\ CM_{x,y,z}(\alpha,\beta) \\ \alpha = [-15:2.5:20] \\ \beta = [-15:2.5:15] \end{aligned} \tag{5}$$

Figure 11 shows the aerodynamic data for the fuselage and ventral fin. Note that contrary to the other components, this data is represented in the wind reference frame. This data is not corrected against CFD, however, the CD0 drag is verified using the OpenVSP ParasiteDrag estimation toolbox. The fuselage and ventral fin CD0 drag is estimated using the Jenkinson method [2]. The CD0 estimate using Jenkinson is 0.00930 and the FlightStrem results show 0.0114. The FlightStream results estimate the CD0 drag to be 23% higher than OpenVSP. The FlightStream results are maintained as they present the more conservative estimate. The table 2 shows the settings used in FlightStream. Note that FlightStream cannot capture the fuselage flow separation and the drag associated with that, hence, the results are not fully accurate at higher sideslips and angles of attack.

Table 2: The FlightStream settings for fuselage and ventral fin data collection

FlightStream version:	#4252022
Viscous-coupling:	Disabled
Drag module:	Reynolds Averaged
S:	29.98 m^2
c:	1.436 m
b:	20.11 m
V:	75 m/s

[2] `http://openvsp.org/wiki/doku.php?id=parasitedrag`

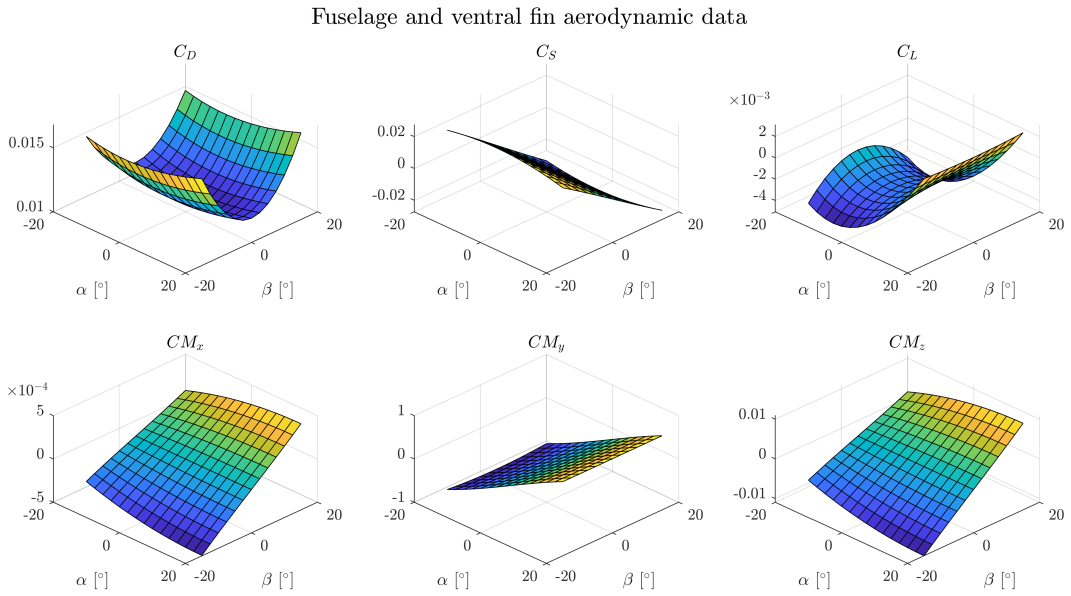


Figure 11: Fuselage and ventral fin aerodynamic data.

2.5 Additional Drag Sources

It is necessary to add an estimate of the additional drag sources that are not contained in the aerodynamic data.

The cooling drag of the vehicle is estimated relatively coarsely by applying an additional drag of 0.015, which is in line with the cooling drag predictions at the cruise phase.

The DEP nacelles are not considered in any of the simulations. The nacelle drag is simply estimated using the OpenVSP Parasite Drag toolbox. The Jenkinson Wing Mounted Nacelle method shows parasite drag of 0.00258 for the nacelle altogether. This drag is added to the baseline drag.

Finally, based on the landing gear state, additional landing gear drag is added. This drag estimate is very rudimentary with the lack of proper CFD analysis. An estimate of the landing gear drag of 0.015 is made based on [2].

3 Aerodynamic Data Interpolation

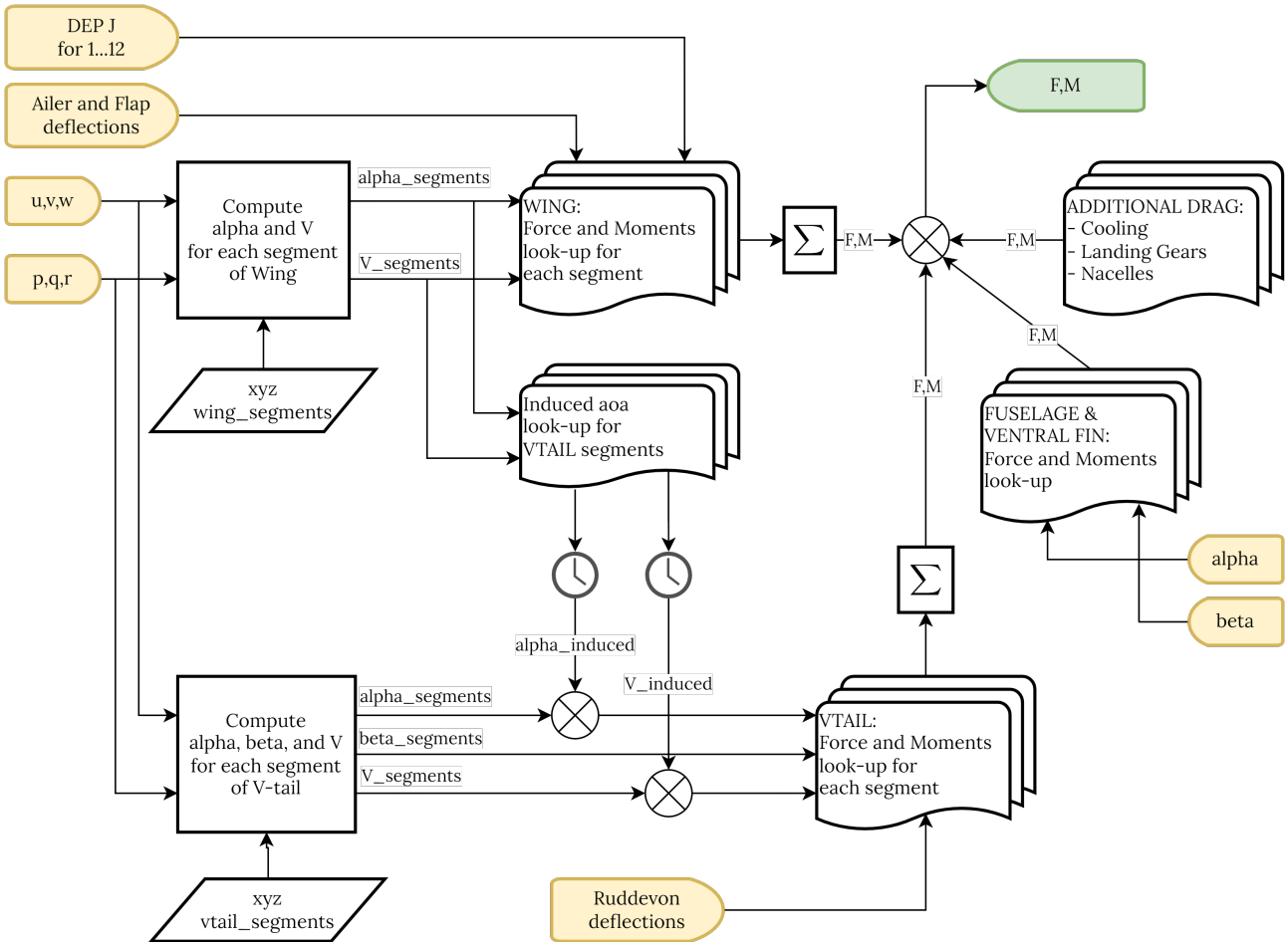


Figure 12: The aerdoynamic data interpolation overview.

Figure 12 shows the overview of the data interpolation process. Note that for the sake of clarity some of the connection are neglected and only the most relevant ones are shown. The interpolation receives the vehicle state (u,v,w,p,q,r,α,β), control surface deflections, and DEP propeller advance ratios as an input.

Firstly, in block number one the local angle of attack and airspeed is calculated for each wing segment reference point. The local angle of attack and airspeed considers the impact of the angular velocity and sideslip.

In block number two the wing forces and moments are interpolated. Note that as only the right wing forces and moments are collected the left wing is interpolated using the same table but changing signs of the outputs F_y, M_x, and M_z. The main source of confusion with the segmented approach is to track how each variable should be correctly assigned to the corresponding segment. The table 3 illustrates how each of the variables are assigned to the segments (the segments can be seen from Figure 7). Note that the numbering goes in a similar order as for DEP propellers. I.e. α_1 is the angle of attack at the left wingtip segment and α_{14} is the angle of attack at the right wingtip segment. Note that as the first segment corresponds to the root segment, the assignments of variables to the left wing segments are "reversed".

In block number three, the induced angle of attack and airspeed on the vertical tailplane due to the main wing is evaluated. The goal of this block is to take into account the downwash generated by the wing on the vertical tailplane. This is possible because when the wing data was collected also the induced velocity at the location of the vertical tailplane was collected. Because each of the wing segments might have its own angle of attack, the wing angle of attack, velocity, and advance ratio the values on the wing are interpolated at the locations of

Table 3: Connection table of wing interpolation

Wing:	Left							Right						
dp_wing_	ROOT					TIP		ROOT					TIP	
Segment:	1	2	3	4	5	6	7	1	2	3	4	5	6	7
α_1							x							
α_2						x								
α_3					x									
α_4				x										
α_5			x											
α_6		x												
α_7	x													
α_8								x						
α_9									x					
α_{10}										x				
α_{11}											x			
α_{12}												x		
α_{13}													x	
α_{14}														x
V_1							x							
V_2						x								
V_3					x									
V_4				x										
V_5			x											
V_6		x												
V_7	x													
V_8								x						
V_9									x					
V_{10}										x				
V_{11}											x			
V_{12}												x		
V_{13}													x	
V_{14}														x
δ_{aL}						x	x							
δ_{aR}													x	x
δ_{fL}	x	x	x	x	x									
δ_{fR}								x	x	x	x	x		
J_1^{DEP}							x							
J_2^{DEP}					x	x								
J_3^{DEP}				x										
J_4^{DEP}			x											
J_5^{DEP}		x												
J_6^{DEP}	x													
J_7^{DEP}								x						
J_8^{DEP}									x					
J_9^{DEP}										x				
J_{10}^{DEP}											x			
J_{11}^{DEP}												x	x	
J_{12}^{DEP}														x

the V-tail segments. This is illustrated by Figure 13. It can be seen that for each V-tail segment the variables at the wing ahead are interpolated. These interpolated variables are then used to interpolate the induced angle of attack and airspeed.

In block number four, it can be seen that delay is added to the induced angle of attack and velocity. This is done to mimic the delay in the wake propagation from the wing to the vertical tailplane. The appropriate delay is determined by the interpolated airpseed at the wing.

In block number five, the local angle of attack, sideslip, and velocity of each V-tail segment is calculated.

In block number six, the V-tail forces and moments are interpolated. This is performed in a similar manner as for the wing, however, it is significantly more simple as there is no need to mirror the data as the data itself contains both left and right V-tail.

Finally, the remaining blocks contain the fuselage and ventral fin forces and moments and the additional drag sources. All the values are summed together and this completes the aerodynamic interpolation.

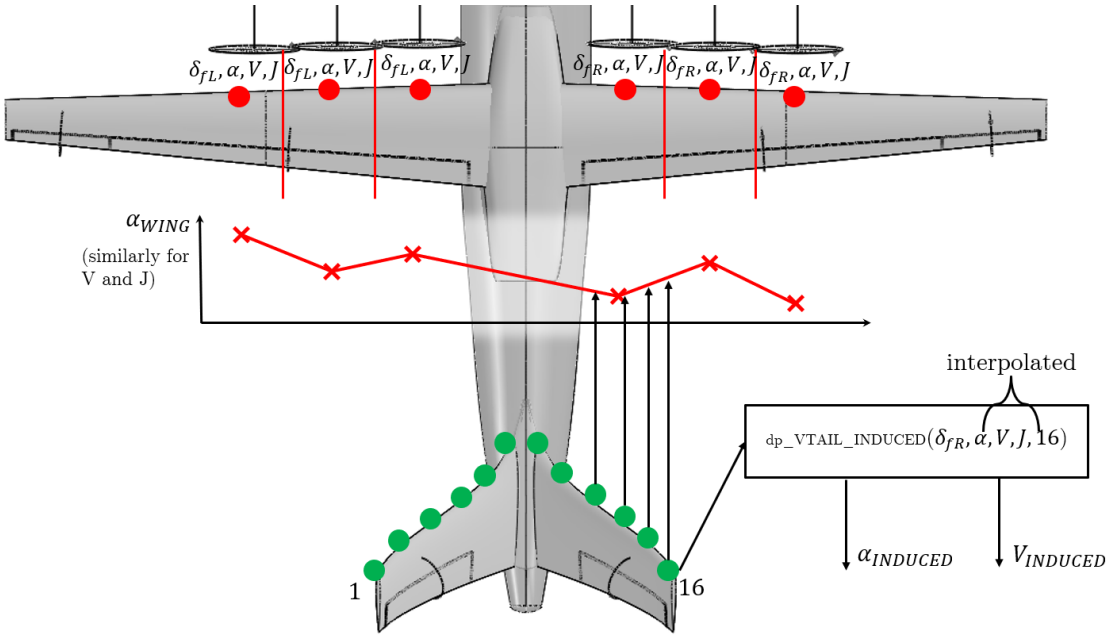


Figure 13: The interpolation of the input variables to the induced angle of attack and velocity data at the locations of the V-tail segments.

4 Vehicle Model Implementation

The flight dynamics simulator is built using Simulink. The model is run using a fixed-step solver with a step size of 500 Hz using the ode1 (Euler) solver. The entire model supports Rapid Acceleration[3] and hence it can be run in real-time even with a mediocre computer.

Note that the simulator should run "out of the box", however, sometimes the already compiled MEX files will not work. Hence, the MEX files can recompiled by first navigating to the *pacer* folder and running:

```
mex sfun_getSystemClockTimeval.c
mex sfun_output_to_console.c
mex sfun_sleep.c
```

And then navigating to the *joystick* folder and running:

```
mex sfun_joyinfoex.c -lwinmm
```

The simulator consists of the following files and folders:

- [file] *UNIFIER.slx*: Contains the Simulink model of the UNIFIER. This is the main model.
- [file] *UNIFIER_LOAD.m*: Is the loading function for the UNIFIER model. The purpose of this file is to load all the required data and variables to the MATLAB workspace.
- [folder] *data*: This folder contains all the "data_packets" (dp_).
  - [file] *dp_VTAIL.mat*: Contains the V-tail aerodynamic data as force and moment look-up tables for each section.
  - [file] *dp_VTAIL_INDUCED.mat*: Contains the influence (down-wash and propeller wake) of the wing on the V-tail as a look-up tables of the induced velocity for each section.
  - [file] *dp_WING.mat*: Contains the wing aerodynamic data as force and moment look-up tables for each section.
  - [file] *dp_FUSE_FIN.mat*: Contains the fuselage and ventral fin aerodynamic data as force and moment look-up tables for each section.
  - [file] *dp_HTU.mat*: Contains the power and torque look-up tables of the HTU propeller used in the HTU model and the operational limits (maximum activity factor w.r.t. airspeed).
  - [file] *dp_DEP.mat*: Contains the thrust and torque coefficient look-up tables of the DEP propeller used in the DEP model and the operational limits (maximum activity factor w.r.t. airspeed).
- [folder] *addons*: Contains Simulink files needed to run the simulation.
  - [folder] *joystick*: Contains the C- and MEX-files used by the custom build sidestick S-Function.
  - [folder] *pacer*: Contains the C- and MEX-files used by a pacer S-function (to keep soft real time).

[3]The rapid accelerator mode creates a Rapid Accelerator standalone executable from your model. This executable includes the solver and model methods, but it resides outside of MATLAB and Simulink. It uses external mode to communicate with Simulink. `https://www.mathworks.com/help/simulink/ug/how-the-acceleration-modes-work.html`

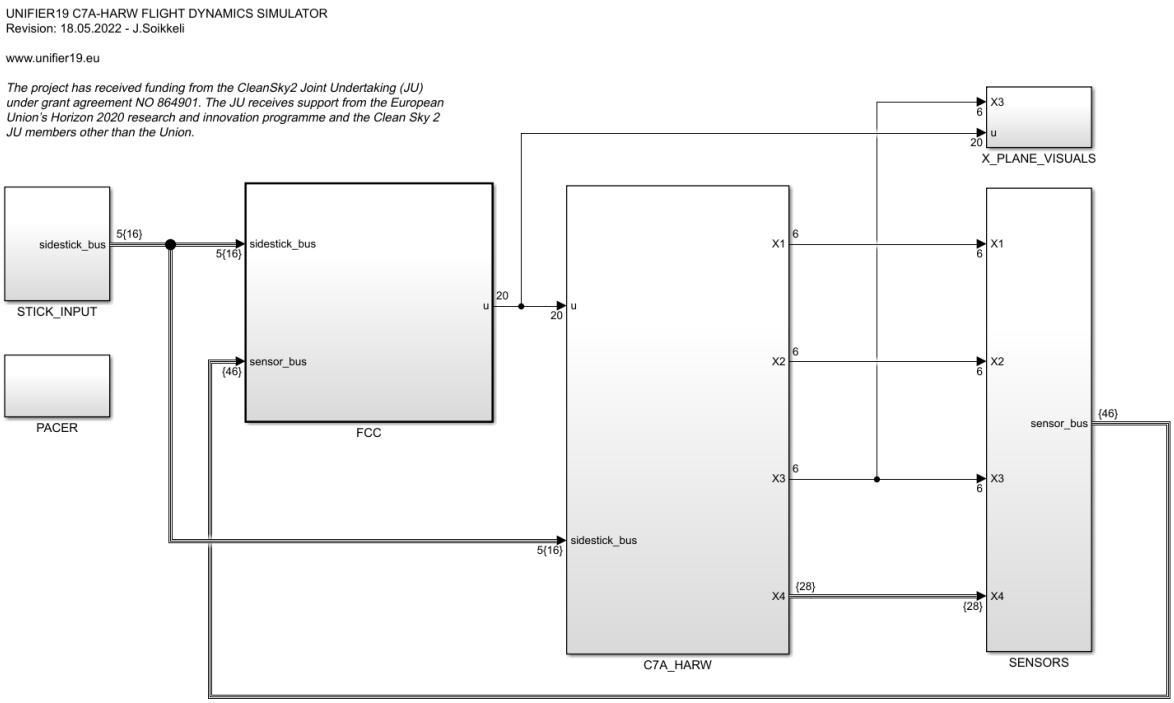


Figure 14: Top level of the UNIFIER19-HARW model

Figure 14 shows the top level of the UNIFIER19 C7A-HARW model (*UNIFIER.slx*). The model consists of the following blocks:

- *STICK_INPUT* subsystem containing the methods used to read values from a sidestick. This subsystem outputs a *sidestick_bus* which contains all the sidestick output raw data: This is 4-axis (values between -1 and 1) and 12 button states (booleans). The core of this subsystem is the *sfun_joyinfoex* S-Function which uses Windows JOYINFOEX library to read sidestick values. This method was developed part of this project. Further information in [3].
- *C7A_HARW* contains the vehicle model. This subsystem contains vehicle aerodynamic model, subsystem models, equations of motion, and the environment model.
- *X_PLANE_VISUALS* contains the methods used to send the aircraft position, orientation and control surface deflection for X-Plane 11 for visualization. The communication to X-Plane is done using UDP packets.
- *PACER* contains the S-function used to keep the simulations in soft real time. This is an external package provided by [4]. The methods support Rapid Accelerator.

4.1 STICK_INPUT

This subsystem contains the methods used to read sidestick inputs. The core of the method is the *sfun_joyinfoex* S-Function. This method is written in C and is based on the JOYINFOEX windows function part of the WinMM library, which was developed as part of this project, and published in [3]. The reason for not using the standard sidestick blocks in the Aerospace Toolbox is because they do not support Rapid Accelerator mode, which is desired for fast real time simulations.

The output of the S-function for the 4-axis (X,Y,Z,R)[4] vary in the range of 0 to 65540. These values are then scaled to range of -1 to 1 for the X, Y, and R. The throttle lever, Z, is scaled to range of 0 to 1. The button outputs are stored as vector of booleans (data type is still double). The axis and buttons are then added to *sidestick_bus*. Figure 15 shows sidestick subsystem.

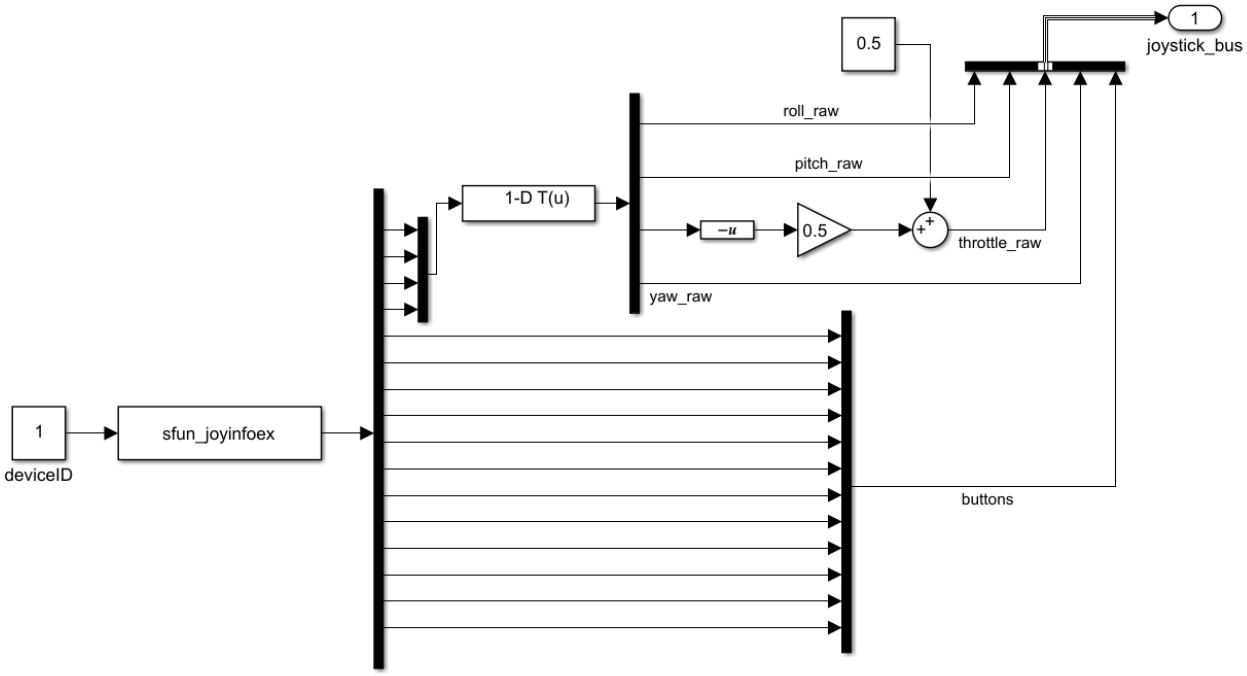


Figure 15: *UNIFIER/STICK_INPUT*

4.2 C7A_HARW

This subsystem contains the vehicle aerodynamic models, the subsystem models, the equations of motion, and the environment models. Hence, it can be seen as the heart of the simulator. Figure 16 shows the subsystem. In principle, the subsystem works by the *EOMAndEnviroment* block generating vehicle trajectory and its derivates (angle of attack, temperature, etc.), which are then used in the subsystem models to generate the forces and moments generated directly by the subsystems and to provide required states of the effectors to the aerodynamic block. Finally, the *Aero* block evaluates the aerodynamic forces and moments which are summed to the subsystem forces and moments. These forces and moments are then used as input in the *EOMAndEnviroment* for the next time step.

[4]X is side-to-side movement, Y is up-to-down movement, Z is throttle lever, and R is twist/yaw.

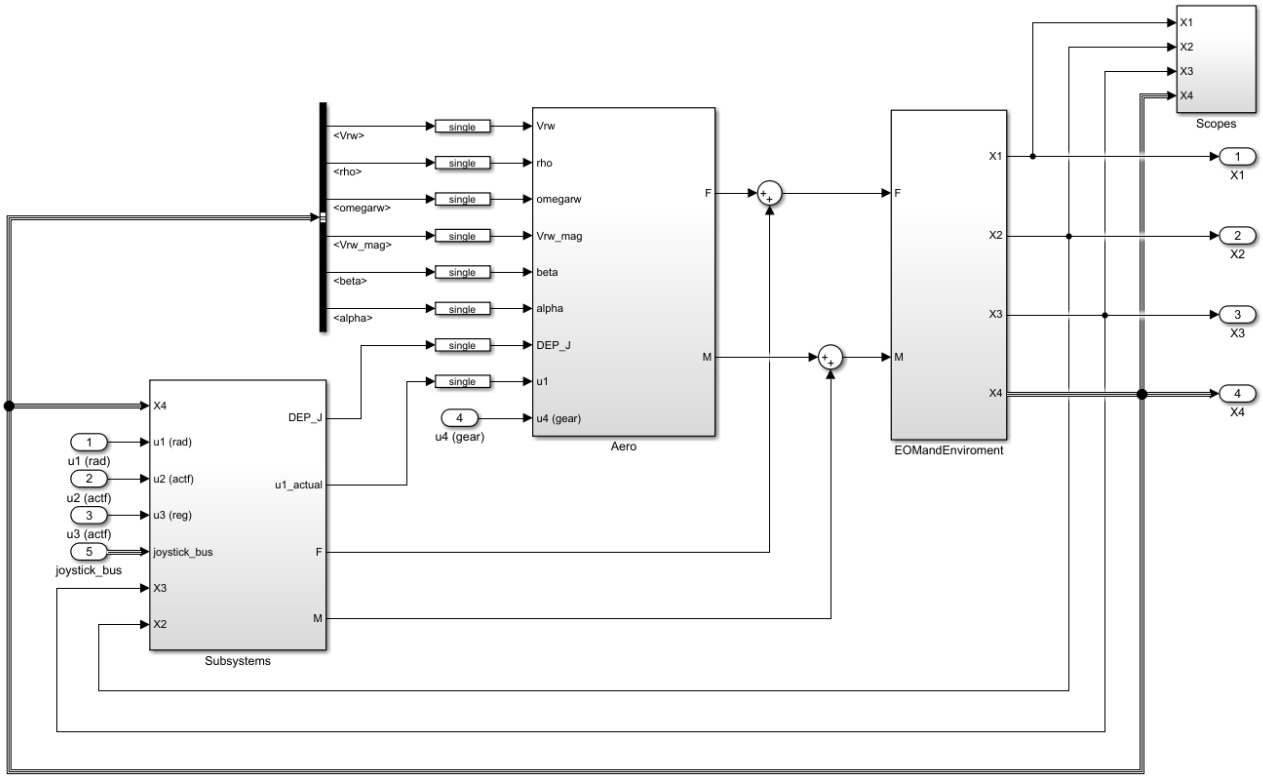


Figure 16: *UNIFIER/C7A_HARW*

4.2.1 EOMAndEnviroment

The *EOMAndEnviroment* subsystem contains the equations of motion and the environment model. For the equations of motion, the *EOM: 6DOF (Euler Angles)* Aerospace Blockset is used. For the environment, the *ISA Atmosphere Model* and simple wind models from the Aerospace Blockset are used. The *EOMAndEnviroment* subsystem outputs are summarized in the table 4.

Table 4: The output of *EOMAndEnviroment* subsystem

Signal	Type	Dimension	Content	Units	Description
X1	mux	6			Acceleration
		3	$(\dot{u}, \dot{v}, \dot{w}) \equiv A_{bb}$	m/s^2	Translational acceleration in body
		3	$(\dot{p}, \dot{q}, \dot{r})$	rad/s^2	Angular acceleration in body
X2	mux	6			Velocity
		3	(u, v, w)	m/s	Translational velocity in body
		3	(p, q, r)	rad/s	Angular velocity in body
X3	mux	6			Position and attitude
		3	(x, y, z)	m	Location of C.G. in earth
		3	(ϕ, θ, ψ)	rad	Euler angles
X4	bus	28			
		3	$(\ddot{x}, \ddot{y}, \ddot{z}) \equiv A_{be}$	m/s^2	Inertial acceleration
		3	$(\dot{x}, \dot{y}, \dot{z})$	m/s	Translational velocity in earth
		3x3	DCM_{be}		Direction cosine matrix earth to body
		3	$(u, v, w)_{rw} \equiv V_{rw}$	m/s	Translational velocity in body w.r.t. wind
		3	$(p, q, r)_{rw} \equiv \Omega_{rw}$	rad/s	Angular velocity in body w.r.t. wind
		1	V_{rw_mag}	m/s	L2-norm of V_{rw} (TAS)
		1	$V_{rw\_mag\_eas}$	m/s	L2-norm of V_{rw} (EAS)
		1	α	rad	Angle of attack
		1	β	rad	Sideslip
		1	T	K	Temperature
		1	P	Pa	Pressue
		1	ρ	kg/m^3	Air density

4.2.2 Subsystems

The *Subsystem* block contains all the vehicle system models. This includes DEP motor and propeller models, gravity, HTU, landing gears, and control surface actuators. The models capture all the system-related dynamics that influence aircraft control. Further expansion of the models would include powertrain models that model the hydrogen system and power delivery.

4.2.3 DEP

The distributed electric propulsion model contains both the electric motor model and the propeller model. The purpose of the DEP model is to evaluate the forces and moments generated directly by the propeller itself, and secondly to provide the advanced ratio at which the propeller operates to the aerodynamic model for evalutation of the forces and moments due to propeller wake interacting with the airframe (aero-propulsive interactions). The same model is used for each of the DEP propulsors.

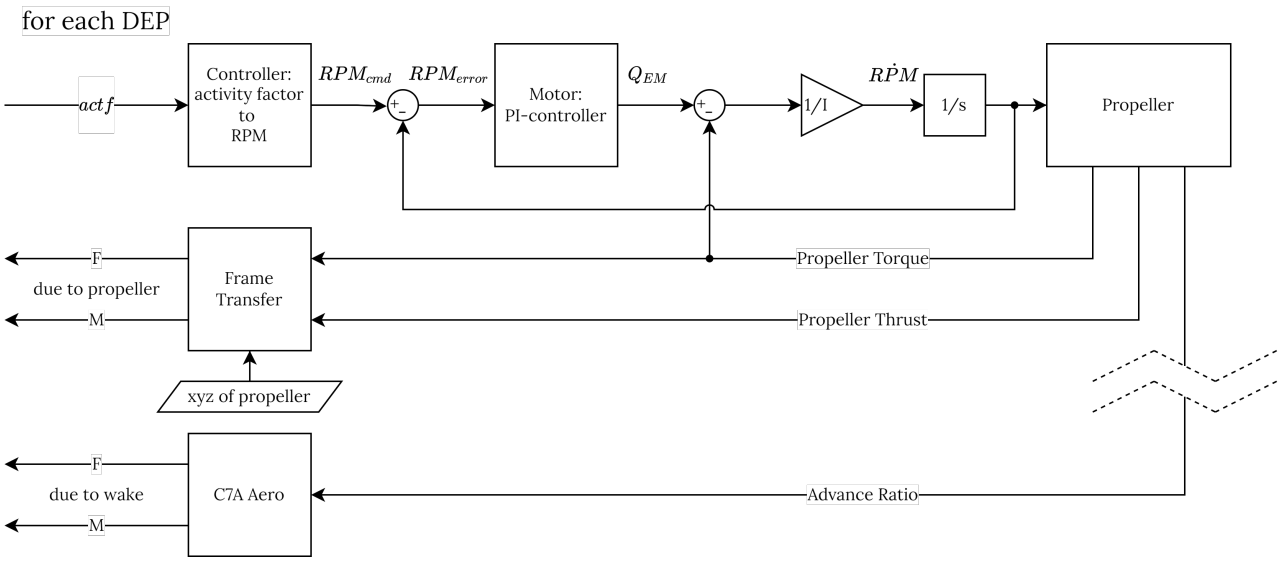


Figure 17: DEP model

Figure 17 illustrates the DEP model. The input to the model is the activity factor which is a unitless value ranging from 0 to 1. The logic behind the activity factor is that a 0 activity factor corresponds to no thrust from the propeller, and 1 corresponds to a fixed maximum thrust. So essentially activity factor is just a scaled thrust request. Naturally, an activity factor of 1 is not achievable in all flight conditions, and hence the CLAWS must be limiting the upper limit on the activity factor. With the activity factor, the region where the propeller is generating negative thrust is avoided, as zero activity factor corresponds to zero thrust (and not zero RPM). The activity factor is converted in the *Controller* to an RPM value. This is achieved by first scaling the activity factor by a gain, so converting the activity factor into thrust, and then by using a lookup table to relate thrust to RPM. This is possible as the propeller operates at a fixed pitch. Hence, for a given airspeed, only one RPM will generate a specific thrust.

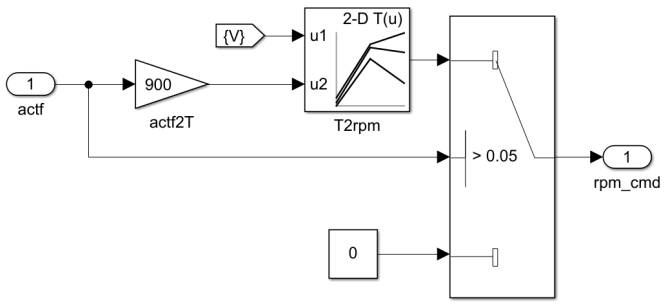


Figure 18: *UNIFIER/C7A_HARW/Subsystems/DEP/DEP/Controller*

Figure 18 shows the conversion from activity factor to RPM. It can be seen that for the propeller to "turn off" correctly, the switch is wrapping any activity factor below 0.05 to zero RPM. This will mean that there will be a discrete jump on RPM when the activity factor crosses 0.05. This is not an issue for the controller, because the time spent in the negative thrust region before the propeller starts to generate thrust is small. This can be seen from Figure 19 which shows the RPM and thrust response of the DEP for a ramp activity factor command at 45 m/s. It can be seen that when the activity factor exceeds the threshold of 0.05 the RPM quickly jumps to an RPM that generates zero thrust. Indeed, it can be seen that it takes a certain amount of time to accelerate the propeller to the correct RPM, and during this time the propeller is firstly generating negative thrust, and

secondly, due to a slight overshoot in the RPM, it is generating "over thrust" momentarily.

The RPM command logic described above is seen as the best compromise of different approaches. If one would control directly RPM then there would exist no jumps in the thrust/RPM, however, then the propeller might operate for longer periods of time in negative/regenerative thrust. This can be seen as beneficial for vehicle control in some applications, however, in general, it is seen that the regenerative propeller would significantly disturb the flow over the wing and cause undesired behaviors. Hence, in this project, it was decided that DEP propellers must always operate in the positive thrust region.

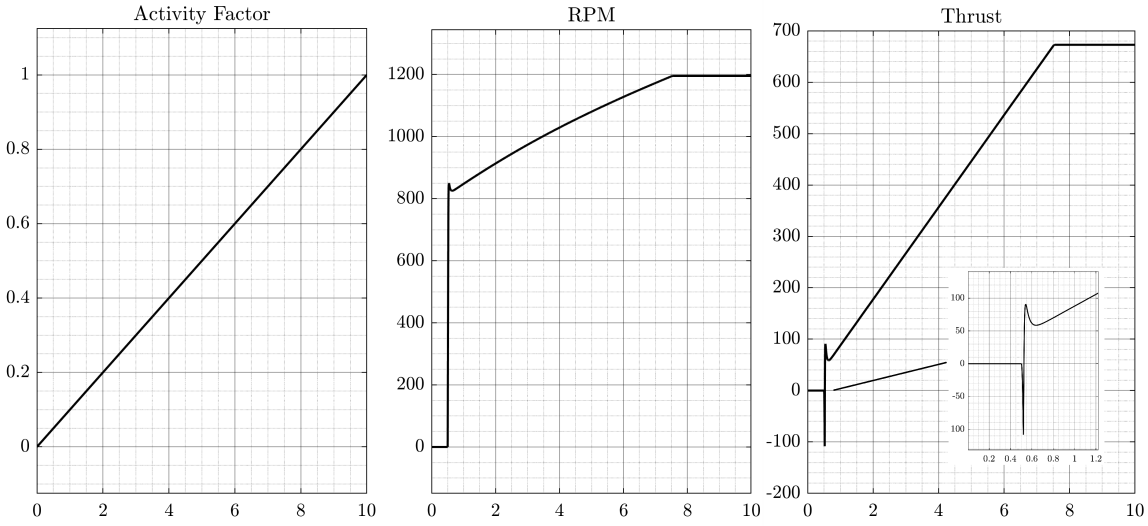


Figure 19: The RPM and thrust response of the DEP for a ramp activity factor command at 45 m/s. The x-axis is time in seconds.

The RPM command generated by the aforementioned controller is fed into the electric motor model. The electric motor is modeled as a PI-controller with saturation. The saturation is set to the maximum torque available from the electric motor. The PI controller parameters are tuned for the propeller inertia to yield a fast, but relatively small overshoot. The dynamics of the model are based on a differential equation:

$$T_{driving} - T_{resisting} = I\dot{\omega} \tag{6}$$

The angular acceleration is integrated to retrieve the rotational speed of the propeller. Naturally, conversion from rad/s to RPM is made (but not depicted in the illustration). The RPM is converted to advance ratio based on the local axial velocity of each DEP propulsor. The local velocity at each DEP is computed with equation 7 in which the cosine is used to account for the propeller frame inclination from the aircraft body axis. Note that only the axial flow component is considered in the propeller-motor model as the propeller data is only for the axial flows. The impact of this assumption (neglecting in-plane propeller forces) is assumed to be relatively small. The propeller thrust and torque coefficients are evaluated from look-up tables with the advance ratio as an input. Finally thrust and torque coefficients are dimensionalized.

$$V_{axial} = cos(5°)[V_{rw}(1) + \Omega_{rw}(3)(xyz_{C.G.}(2) - xyz_{DEP}(2))] \tag{7}$$

Based on the propeller locations and inclination, frame transfer is performed to translate the thrust and torque from the propeller frame to the aircraft's center of gravity. In order to evaluate the impact of the propeller wake over the wing, the advance ratio is an output of the DEP model. This advance ratio is used in the aerodynamics model to define the forces and moments due to the aero-propulsive interactions.

Figure 20 illustrates the upper limit on the activity factor based on airspeed.

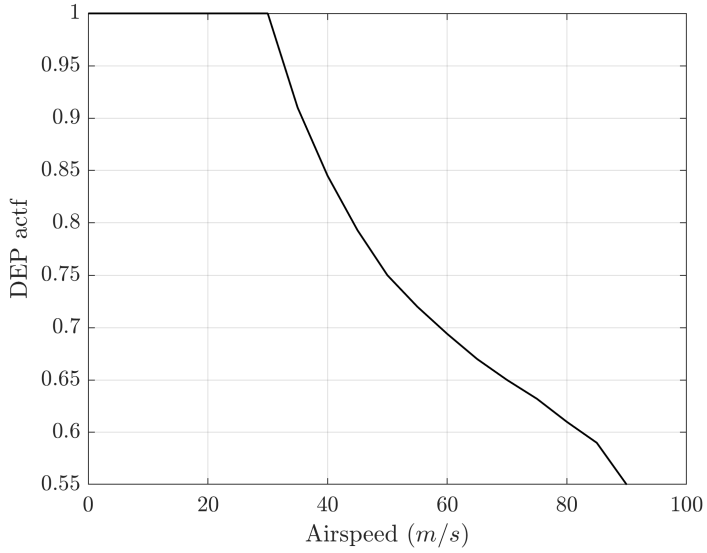


Figure 20: The upper limit of the activity factor based on airspeed.

4.2.4 HTU

The HTU is also controlled using the activity factor, which similarly to DEP, maps to a certain thrust range. In contrast to the DEP, the HTU activity factor is allowed to be negative to enable the use of the propeller for drag generator. This is desirable on approach conditions to allow DEP to operate at higher activity factors, hence, increasing the aero-propulsive interactions. Additionally, the HTU propeller is variable pitch, hence there does not exist a specific RPM corresponding to a specific thrust. Hence in order to model the HTU, the RPM problem must be solved first.

Figure 22 shows the resulting power and operating bounds. Figure 23 illustrates the process described above to derive the power data and operating bounds. The optimum power RPM strategy is designed for the HTU, which means that the RPM that results in the best efficiency is selected for each operating point. This is achieved by using X-rotor to generate a 4D look-up table containing HTU torque as a function of airspeed, thrust, and RPM. From the table, the RPM that corresponds to minimum power is selected. This is also true for regenerative mode as maximum energy, so most negative torque, is wanted from the propeller. Based on these selected RPMS, an optimum operating RPM is known. This optimum RPM control strategy is shown in Figure 21.

Based on the optimum RPM, the RPM dimension is removed from the HTU multi-dimensional data as it is assumed that the HTU can instantly operate at the optimum RPM. This might not actually be correct because during transients the RPM might take longer to adjust to different thrust requests, however, here the assumption is made that RPM control is instantaneous and hence it is assumed that for given thrust and airspeed the HTU operates always according to the optimum RPM strategy.

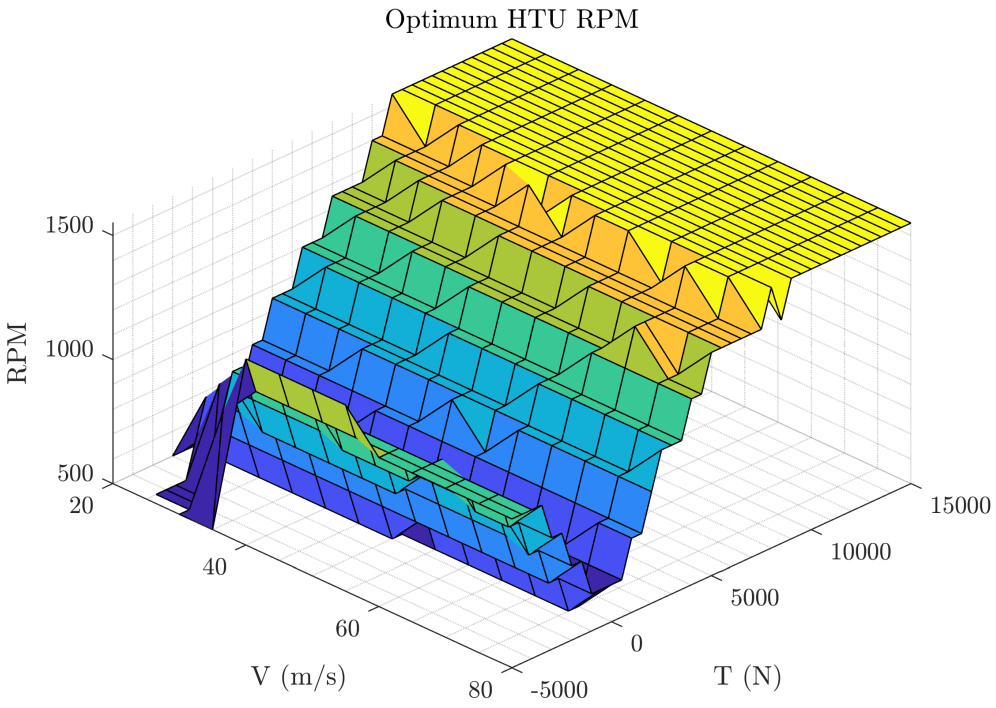


Figure 21: The optimum RPM for different operating conditions for the HTU propeller.

The torque look-up table is used to make power look-up table (by multiplying the torque values with the corresponding optimal rotational speed). This will result in the power look-up table. Based on the operating limits of the HTU motor (limited in power) the operating limits are defined.

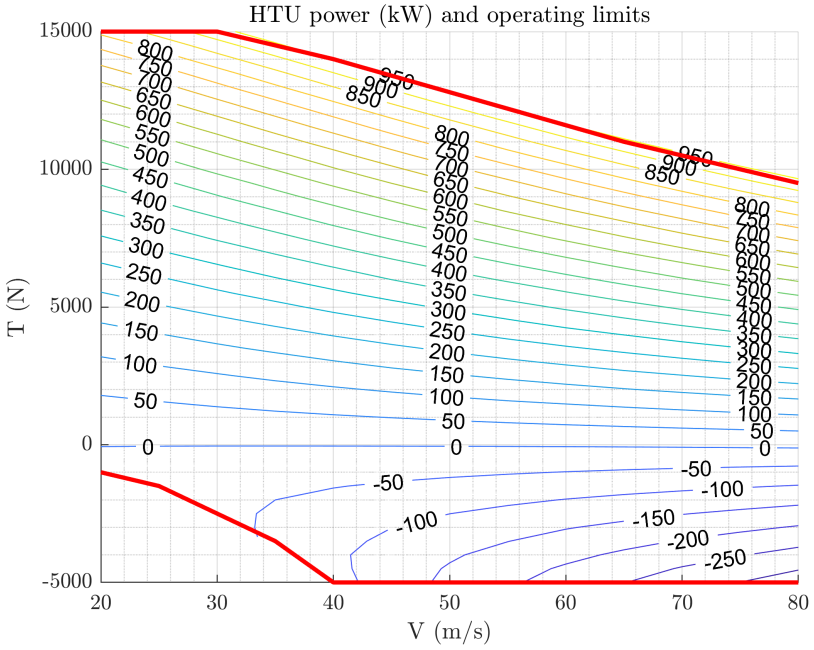


Figure 22: HTU power (kW) and the operating bounds

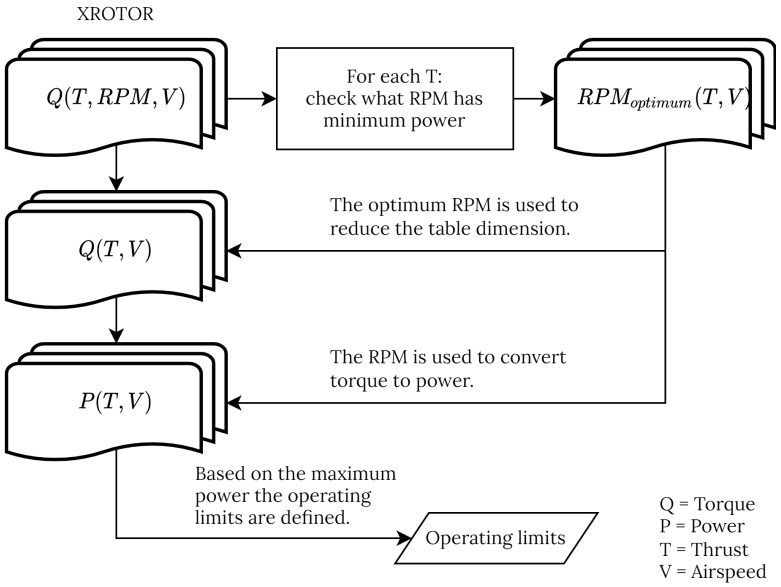


Figure 23: The process visualized to derive the operating limits and power look-up table

Figure 24 shows the HTU model. It can be seen that the dynamics of the HTU are modeled as a linear second-order model with a natural frequency of 8 rad/s and damping of 1.1. These values are based on previous experience with high-power electric motors with variable pitch propellers.

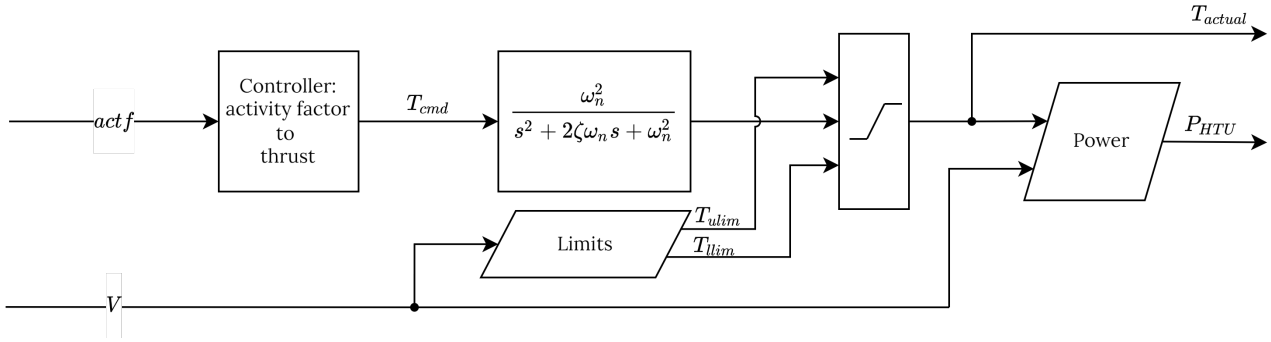


Figure 24: The HTU model

4.2.5 Actuators

The control surface actuators are modeled as non-linear second-order models. The model is made using the standard actuator model part of the Aerospace Toolbox blockset. The parameters of the actuator model are listed in table 5. The parameters of the actuators are based on an estimate of the performance of advanced electric duplex actuators.

Table 5: Non-linear second order actuator model parameters

Natural frequency	50	rad/s
Damping ratio:	0.8	
Maximum deflection:	+25 (ail, flap)	deg
	+15 (rude)	deg
Minimum deflection:	-25 (ail)	deg
	0 (flap)	deg
	-30 (rude)	deg
Rate limit:	120 (ail, rude)	deg/s
	60 (flap)	deg/s

4.3 X_PLANE_VISUALS

This subsystem is responsible for sending the data from Simulink to X-Plane 11 for visualization. The subsystem was developed as part of this project and it uses UDP data packets to send the vehicle location and orientation, control surface deflections, and HTU and DEP propulsor states. The IP address and port number in Simulink must match the IP and port number in X-Plane 11. The X-Plane 11 model of the vehicle is custom designed as part of this project.

The vehicle orientation is communicated to X-Plane 11 by sending *VEHX* data-packet. The data packet is 45 bytes long. The data packet is shown in table 6. The control surface deflection data is communicated by setting the corresponding data reference (dref)[5]. The UDP packet (called DREF) to do this is a generic one that can be used to set any dref value. The dref data packet is 509 bytes long with the message as long as required, and the rest of the message is padded with zeros. The format of the DREF data packet is shown in table 7.

Note that before the control surface data packets can be sent (or they can be sent but they won't work) the dref *'sim/operation/override/override_flightcontrol'* must be set to 1. This is done in the Simulink model by the triggered subsystem which is triggered once at the start of the simulation.

Table 6: VEHX UPD data packet format containing the vehicle location and orientation information

Bytes	Length	Content	Comment
1:5	5 bytes	uint8('VEHX0')	[86 69 72 88 48]
6:9	4 bytes	uint8(zeros(1,4))	
10:17	8 bytes	typecast(lat,'uint8')	double precision
18:25	8 bytes	typecast(long,'uint8')	double precision
26:33	8 bytes	typecast(h,'uint8')	double precision
34:37	4 bytes	typecast(psi,'uint8')	single precision
38:41	4 bytes	typecast(the,'uint8')	single precision
42:45	4 bytes	typecast(phi,'uint8')	single precision

[5] `https://developer.x-plane.com/datarefs/`

Table 7: DREF UPD data packet format for setting any dref field.

Bytes	Length	Content	Comment
1:5	4 bytes	uint8('DREF0')	[68 82 69 70 48]
6:9	4 bytes	typecast(value,'uint8')	payload as a single precision
10:x		uint8(str)	dref name
x+1:509		0	padding

References

[1] J. Soikkeli, "Vertical tail reduction through differential thrust: An initial assessment of aero-propulsive effects on lateral-directional stability and control in engine inoperative conditions," 2020.

[2] J. Williams, J. Callaghan, D. Foster, G. Bowes, C. Layman, and W. Loeve, "Prediction methods for aircraft aerodynamic characteristics," ADVISORY GROUP FOR AEROSPACE RESEARCH AND DEVELOPMENT NEUILLY-SUR-SEINE (FRANCE), Tech. Rep., 1974.

[3] J. Soikkeli, "S-function for reading joystick values on simulink," https://www.mathworks.com/matlabcentral/fileexchange/111265-s-function-for-reading-joystick-values-on-simulink, 2022, [MATLAB Central File Exchange. Retrieved May 18, 2022].

[4] M. Compere, "Soft real time block for pacing simulink simulations," https://www.mathworks.com/matlabcentral/fileexchange/67520-soft-real-time-block-for-pacing-simulink-simulations, 2022, [MATLAB Central File Exchange. Retrieved May 19, 2022].