MAVIS Documentation and User Manual

Welcome to MAVIS

MAVIS is a versatile flight simulation tool for simulating the physics of air vehicles. It is a stand-alone physics simulation package designed to interface with other stand-alone graphics and controls software. The physics is flexible enough to model a large range of aircraft configurations including fixed-wing aircraft, rotorcraft, rockets, and many other geometries.

Select the “Show document outline”  icon on the left to view a Table of Contents.

1. Installation

A. From an Executable

An executable for Windows or Mac can be requested by sending an email to doug.hunsaker@usu.edu.

The executable will be distributed in a zipped folder that includes the executable along with two additional folders. One folder is a test directory that contains test cases outlined in Testing the Installation. The second folder contains several example cases outlined in Examples.

The Windows executable is currently built using Cygwin, and therefore will require Cygwin to be installed on the Windows machine in order to run. It is helpful to copy the executable into your user directory path such that it is accessible from the Cygwin command prompt in any directory.

B. From Source Code

The code can be downloaded from github using this link. Build the executable by running the build.sh script. Before running the script, it must be an executable. To make the build.sh script executable, open a terminal window, navigate to the MAVIS folder, and type

On a Mac: `chmod +x build.sh`

On Windows:

As a last step, the build.sh script attempts to copy the executable to /usr/local/bin so that it can be accessed from anywhere on the machine. In order for that to work correctly on a mac, the build script must be run with root permissions:

```
sudo ./build.sh
```

You can avoid needing to type in credentials each time by opening the shell with root privileges by typing

```
sudo -s
```

before running the build script. If you don't run the build script with root permissions, the executable will still build, but will only be available in the /bin directory. The build script for the main code also builds a test executable stored in the testing folder.

Note: If you get the error `$'\r': command not found in build script`, try changing the end-line characters using the command `sed -i 's/\r$//' build.sh`

C. Testing the Installation

The installation can be tested by navigating to the test folder and running the command

```
./test test.json
```

This will run a series of tests listed in the test.json file. Additional tests can be added by creating new directories, one for each test, and adding the name of the directory in the json file along with the output file to be compared after running Mavis. The test executable always looks for a file in the directory called expected_result.csv against which to compare the output file.

2. Running MAVIS

Once compiled, the code can be run by opening a command prompt in the directory in which the executable is stored and typing

```
./mavis <input_filename>
```

where the input_filename is the name of a .json file that contains the instructions for running the code. Or, if you moved the executable to a location accessible in your path, you can execute the code from any directory by typing

```
mavis <input_filename>
```

Currently there is no GUI available for MAVIS.

3. Input File Structure

MAVIS uses a JSON input file to read in all commands for the code. The JSON file structure allows for commands and information within the JSON file to be nested within JSON dictionaries. The top-level JSON file must contain the following dictionaries:

```
{   "simulation": {...},
    "atmosphere": {...},
    "vehicle": {...},
    "view": {...}
}
```

Any dictionary within the JSON file can read from another JSON file using the command

```
"filepath" : <path to json dictionary file>
```

For example, if all of the information required in the simulation dictionary were stored in a file named “simulation.json” one directory higher than the main file path, the main file would have an entry for simulation as follows:

```
{   "simulation": {"filepath" : "../simulation.json"},
    "atmosphere": {...},
    "vehicle": {...},
    "view": {...}
}
```

This technique can be used for any dictionary or sub-dictionary in the input file structure.

A. Simulation Dictionary

The simulation dictionary contains the following:

Key	Value Type	Allowed Values	Required / Optional	Default Value	Description
name	string	arbitrary	optional	MySim	The name of the simulation.
begin_time[sec]	float	arbitrary	optional	0.0	Beginning time of the simulation.
end_time[sec]	float	arbitrary	Required		Ending time of the simulation.
time_step[sec]	float	>= 0	optional	0.0	Time step used in simulation. If 0, the simulation is set to real-time and will use a time

Key	Value Type	Allowed Values	Required / Optional	Default Value	Description
					step as small as possible to make the simulation run in real-time.
ground_altitude[ft]	float	arbitrary	optional	0.0	Altitude of the ground plane. This is used for collision and landing.
states_print_rate[hz]	float	> 0	optional	1.0	Rate at which the states should be printed to the console during execution.
states_save_rate[hz]	float	> 0	optional	1.0	Rate at which the states should be saved to a file during execution.
pulse_rate[hz]	float	> 0	optional	1.0	Rate at which simulation execution speed should be printed to the console during execution.
connection	dictionary		optional		See Connections Dictionary.

Connections Dictionary

The simulation→connections dictionary contains a list of dictionaries with arbitrary names. Each name must start with the string “send_” or “receive_”. For example, the name of one connection could be “send_to_graphics” or “receive_from_controller”. The first word delimited by the underscore tells the connection whether it should be sending information or receiving information for that connection. Each sub-dictionary contains the following:

Key	Value Type	Allowed Values	Required / Optional	Default Value	Description
array_length	integer	> 0	optional		Length of array of **floats** to be passed for message type -1. Required if message_type = -1.
std_dev	float array of length 1 or array_length	arbitrary	optional	[0.0]	Standard deviation of random numbers to be added to the array when sending/receiving. If a single value is given in the array, this same standard deviation is applied to all

Key	Value Type	Allowed Values	Required / Optional	Default Value	Description
					values in the array for this connection. If more than one value is given in the array, the array must have the same length as given by array_length, and each value represents the standard deviation to be added to each corresponding value in the value array. This is helpful for testing how noisy data/connections might affect the solution.
port_type	string	UDP, file	required		Whether to use UDP for communication or send/receive from a file.
filename	string	arbitrary ending in .csv	optional		Required for port_type = file.
file_header	string	arbitrary	optional	“Time[s], variables →”	Header to be used when sending data to a CSV file.
port_ID	integer	> 0	optional		Port number to be used for the connection. Required for port_type = UDP.
IP	IP Address		optional	127.0.0.1	IP address where the port should be accessed.
refresh_rate[hz]	float	> 0	optional	100	Rate at which the connection should be read/written during execution.
wait_for_data	bool	true, false	optional	false	A flag to tell any receiving port to wait for data on the port before returning. If true, the code will not progress until data is read from the port. If false, the code will continue to progress and simply use the last data read from the port. This option is only available for connections used for receiving.
verbose	bool	true, false	optional	false	A flag to output verbose information during runtime.

More on how to interface with MAVIS is given in the Interfacing with MAVIS section.

B. Atmosphere Dictionary

The simulation dictionary contains the following:

Key	Value Type	Allowed Values	Required / Optional	Default Value	Description
constant_wind[ft/s]	float array of length 3	arbitrary	optional	[0.0, 0.0, 0.0]	Array containing the direction and magnitude of constant wind.
gusts	dictionary		optional		See Gusts Dictionary.

Gusts Dictionary

The atmosphere→gusts dictionary contains the following:

Key	Value Type	Allowed Values	Required / Optional	Default Value	Description
type	string	none, dryden_beal	optional	none	The type of gusts to be included in the simulation.
intensity	string	light, moderate, severe	optional	light	The level of turbulence intensity to be used in the simulation.
ramp_in[sec]	float	> 0	optional	1.0	Time required to ramp in turbulence so there is not a step change in atmospheric conditions. This time applies at the beginning of the simulation for type = “dryden_beal”, or at the time an aircraft enters the bounding box of a turbulence database for type = “database”.
seed	string	auto, repeatable	optional	auto	How to seed the random number generator. If auto, the computer time is used to seed

Key	Value Type	Allowed Values	Required / Optional	Default Value	Description
					the generator. If repeatable, the value 0 is used to seed the generator, so the solutions are repeatable.
sample	dictionary		optional		See Sample Dictionary.

Sample Dictionary

The atmosphere→gusts→sample dictionary contains the following:

Key	Value Type	Allowed Values	Required / Optional	Default Value	Description
save_filename	string	arbitrary	optional	sample_gust_output.csv	The type of gusts to be included in the simulation.
number_of_points	integer	> 0	optional	100	Number of points to sample.
time[s]	float array of length 2	arbitrary	required		Beginning and ending sample times as an array.
x[ft]	float array of length 2	arbitrary	required		Beginning and ending values for x as an array.
y[ft]	float array of length 2	arbitrary	required		Beginning and ending values for y as an array.
z[ft]	float array of length 2	arbitrary	required		Beginning and ending values for z as an array.

C. Vehicle Dictionary

The vehicle dictionary contains the following:

Key	Value Type	Allowed Values	Required / Optional	Default Value	Description
properties	dictionary		required		See Properties Dictionary.
initial	dictionary		required		See Initial Dictionary.

Properties Dictionary

The vehicle→properties dictionary contains the following:

Key	Value Type	Allowed Values	Required / Optional	Default Value	Description
airfoils	dictionary		optional		See Airfoils Dictionary. Required if referenced by a wing or rotor component.
propulsion	dictionary		optional		See Propulsion Dictionary. Required if referenced by a rotor component.
components	dictionary		required		See Components Dictionary.
control_effectors	dictionary		optional		See Control Effectors Dictionary.

Airfoils Dictionary

The vehicle→properties→airfoils dictionary contains a list of airfoils and their thickness parameters. Any number of airfoils can be included in this dictionary. Each airfoil is a component of the dictionary with a key name specified by the user. Each sub-dictionary contains the thickness distribution parameters for a single airfoil. Two airfoil thickness functions are currently supported, although others can easily be added. These are a NACA 4-digit thickness distribution function, or a diamond airfoil thickness distribution.

If the key name "a0" appears in the dictionary, the code defaults to the NACA 4-digit distribution and requires the following parameters.

Key	Value Type	Allowed Values	Required / Optional	Default Value	Description
a0	float	arbitrary	required		First term in the NACA 4-digit airfoil thickness distribution.
a1	float	arbitrary	required		Second term in the NACA 4-digit airfoil thickness distribution.
a2	float	arbitrary	required		Third term in the NACA 4-digit airfoil thickness distribution.
a3	float	arbitrary	required		Fourth term in the NACA 4-digit airfoil thickness distribution.
a4	float	arbitrary	required		Fifth term in the NACA 4-digit airfoil thickness distribution.

If the key name "a0" does not appear in the dictionary, the code defaults to a diamond airfoil, which instead requires the following parameter to be specified:

Key	Value Type	Allowed Values	Required / Optional	Default Value	Description
xm/c	float	arbitrary	required		The x-location of airfoil maximum thickness in percent chord (x/c) for a diamond airfoil.

Propulsion Dictionary

The vehicle→properties→propulsion dictionary contains a list of propulsion elements and their parameters. Any number of propulsion elements can be included in this dictionary. Each is a component of the dictionary with a key name specified by the user. Each sub-dictionary contains the following:

Key	Value Type	Allowed Values	Required / Optional	Default Value	Description
type	string	polynomial, T=f(V), default	required		Defines the type of propulsion unit.

If type is “polynomial”, the following parameters can be specified:

Key	Value Type	Allowed Values	Required / Optional	Default Value	Description
CT	float array of arbitrary length	arbitrary	required		The components of this array specify the polynomial coefficients to be used to compute the thrust as a function of advance ratio.
CPb	float array of arbitrary length	arbitrary	optional	[0.0]	The components of this array specify the polynomial coefficients to be used to compute the break power as a function of advance ratio.
CN,alpha[1/rad]	float array of arbitrary length	arbitrary	optional	[0.0]	The components of this array specify the polynomial coefficients to be used to compute the normal force gradient per radian as a function of advance ratio.
Cn,alpha[1/rad]	float array of arbitrary length	arbitrary	optional	[0.0]	The components of this array specify the polynomial coefficients to be used to compute the yawing-moment gradient per radian as a function of advance ratio.

If type is “T=f(V)”, the following parameters must be specified:

Key	Value Type	Allowed Values	Required / Optional	Default Value	Description
T_coefficients	float array of arbitrary length	arbitrary	required		The components of this array specify the polynomial coefficients to be used to compute the thrust as a function of velocity.
a	float	arbitrary	required		Exponent to scale the density ratio by.

If type is “default”, the propulsion element uses properties for a family of propellers as outlined by Hunsaker [1]. In this case, the following parameter must be specified:

Key	Value Type	Allowed Values	Required / Optional	Default Value	Description
pitch_to_diameter	float	arbitrary	required		The pitch-to-diameter ratio of the rotor.

Components Dictionary

The vehicle→properties→components dictionary contains a list of dictionaries with arbitrary names, each for a different component of the vehicle. Any number of elements can be included in this dictionary. Each key name is specified by the user. Each sub-dictionary contains the following:

Key	Value Type	Allowed Values	Required / Optional	Default Value	Description
type	string	cuboid, cylinder, sphere, wing, rotor, engine, custom	required		This parameter specifies the type of component. Each component requires slightly different parameters to be specified as outlined below.
location[ft]	float array of length 3	arbitrary	optional	[0.0, 0.0, 0.0]	Location of the component coordinate origin relative to the aircraft coordinate origin in aircraft coordinates. Components refer to x, y, and z locations.
orientation[deg]	float array of length 3	arbitrary	optional	[0.0, 0.0, 0.0]	Orientation of the component coordinate system relative to the aircraft coordinate system in terms of Euler angles. Components refer to ϕ, θ, and ψ angles in the traditional Euler angle formulation and rotation matrix [R] given by Hunsaker [1]. Note: This parameter is not used for type “wing”.
include_aero	boolean	true, false	optional	false	If true, the aerodynamics of the component are included in the aerodynamic model for the entire aircraft.
weight[lbf]	float	arbitrary	optional	0.0	Weight of the component.
mass[slug]	float	arbitrary	optional	0.0	Mass of the component.
density[slug/ft^3]	float	arbitrary	optional	0.0	Density of the component.

Cuboid

If type is “cuboid” the following parameters can be specified:

Key	Value Type	Allowed Values	Required / Optional	Default Value	Description
length[ft]	float array of length 3	arbitrary	required		An array containing the three lengths that define the external size of the cuboid in feet. The array components correspond to the x, y, and z lengths of the cuboid in the cuboid coordinate system.
length_inner[ft]	float array of length 3	arbitrary	optional	[0.0, 0.0, 0.0]	An array containing the three lengths that define the internal size of the cuboid in feet, for the case of a hollow cuboid. The array components correspond to the x, y, and z lengths of the cuboid in the cuboid coordinate system.

Cylinder

If type is “cylinder” the following parameters can be specified:

Key	Value Type	Allowed Values	Required / Optional	Default Value	Description
length[ft]	float	arbitrary	required		The length of the cylinder.
radius[ft]	float	arbitrary	required		The external radius of the cylinder.
radius_inner[ft]	float	arbitrary	optional	0.0	The internal radius of the cylinder for the case of a hollow cylinder.

Sphere

If type is “sphere” the following parameters can be specified:

Key	Value Type	Allowed Values	Required / Optional	Default Value	Description
radius[ft]	float	arbitrary	required		The external radius of the sphere.
radius_inner[ft]	float	arbitrary	optional	0.0	The internal radius of the sphere for the case of a hollow sphere.

Wing Segment

If type is “wing” the following parameters can be specified:

Key	Value Type	Allowed Values	Required / Optional	Default Value	Description
side	string	left, right, both	required		Specifies how the wing will be treated, as a right wing, left wing, or mirrored (both) wing.
span[ft]	float	arbitrary	required		The span of the wing segment.
sweep[deg]	float	arbitrary	optional	0.0	The quarter-chord sweep angle of the wing segment.
dihedral[deg]	float	arbitrary	optional	0.0	The dihedral angle of the wing segment.
chord[ft]	float array of length 2	arbitrary	required		A float array with two components specifying the root and tip chord of the wing segment in feet. The first entry corresponds to the root chord, and the second entry corresponds to the tip chord.
thickness[%]	float array of length 2	arbitrary	required		A float array with two components specifying the root and tip airfoil thickness of the wing segment in percent chord. The first entry corresponds to the root airfoil thickness, and the second entry corresponds to the tip airfoil thickness.
airfoil	string	Must match one entry in the airfoils dictionary	required		A string containing the name of the airfoil to be used to compute the airfoil thickness distribution. Information about the airfoil thickness distribution is contained in the Airfoils Dictionary. The name of the airfoil used here must match one in the airfoils dictionary.
aerodynamics	dictionary				See Aerodynamics Dictionary.

Rotor

If type is “rotor” the following parameters can be specified:

Key	Value Type	Allowed Values	Required / Optional	Default Value	Description
rotation	string	RH, LH	optional	RH	Denotes right-hand or left-hand rotation of the rotor with the thumb pointed in the opposite direction of the thrust.
blade_count	integer	> 0	required		The number of blades on the rotor.
rotor_diameter[ft]	float	> 0	required		The diameter of the rotor.
hub_diameter[ft]	float	> 0	required		The diameter of the rotor hub.
hub_height[ft]	float	> 0	required		The height or thickness of the rotor hub.
chord[ft]	float array of length 2	arbitrary	required		A float array with two components specifying the root and tip chord of a single rotor blade in feet. The first entry corresponds to the root chord, and the second entry corresponds to the tip chord.
thickness[%]	float array of length 2	arbitrary	required		A float array with two components specifying the root and tip airfoil thickness of a single rotor blade in percent chord. The first entry corresponds to the root airfoil thickness, and the second entry corresponds to the tip airfoil thickness.
airfoil	string	Must match one entry in the airfoils dictionary	required		A string containing the name of the airfoil to be used to compute the airfoil thickness distribution. Information about the airfoil thickness distribution is contained in the Airfoils Dictionary. The name of the airfoil used here must match one in the airfoils dictionary.

propulsion	string	Must match one entry in the propulsion dictionary	optional		The name of the propulsion element to be used to compute the propulsion properties of the rotor. Information about the propulsion properties is contained in the Propulsion Dictionary. The name of the propulsion element used here must match one in the propulsion dictionary. Required if include_aero is true.

Engine

If type is “engine” the following parameters can be specified:

Key	Value Type	Allowed Values	Required / Optional	Default Value	Description
propulsion	string	Must match one entry in the propulsion dictionary	optional		The name of the propulsion element to be used to compute the propulsion properties of the rotor. Information about the propulsion properties is contained in the Propulsion Dictionary. The name of the propulsion element used here must match one in the propulsion dictionary. Required if include_aero is true.

Custom

If type is “custom” the following parameters can be specified:

Key	Value Type	Allowed Values	Required / Optional	Default Value	Description
local_CG[ft]	float array of length 3	arbitrary	optional	[0.0, 0.0, 0.0]	The location of the center of gravity in the component coordinate system.
local_inertia_reference_location[ft]	float array of length 3	arbitrary	optional	[0.0, 0.0, 0.0]	The reference location about which the inertia tensor is specified by the user.
local_aero_reference_location[ft]	float array of length 3	arbitrary	optional	[0.0, 0.0, 0.0]	The reference location about which the aerodynamic information is specified by the user.
inertia	dictionary		required		See Inertia Dictionary.
angular_momentum[slug-ft^2/s]	float array of length 3	arbitrary	optional	[0.0, 0.0, 0.0]	Angular momentum vector and magnitude in the component coordinate system.
aerodynamics	dictionary		optional		Required if include_aero is true. See Aerodynamics Dictionary.

Inertia Dictionary

The vehicle→properties→components→inertia dictionary contains either the following:

Key	Value Type	Allowed Values	Required / Optional	Default Value	Description
Ixx[slug-ft^2]	float	arbitrary	required		Ixx moment of inertia component.
Iyy[slug-ft^2]	float	arbitrary	required		Iyy moment of inertia component.
Izz[slug-ft^2]	float	arbitrary	required		Izz moment of inertia component.
Ixy[slug-ft^2]	float	arbitrary	optional	0.0	Ixy moment of inertia component.
Ixz[slug-ft^2]	float	arbitrary	required		Ixz moment of inertia component.
Iyz[slug-ft^2]	float	arbitrary	optional	0.0	Iyz moment of inertia component.

Or, it can contain the same information as a Generic Model Dictionary.

Aerodynamics Dictionary

The vehicle→properties→components→aerodynamics dictionary contains the same information as a Generic Model Dictionary. Additionally, it contains the following:

Key	Value Type	Allowed Values	Required / Optional	Default Value	Description
reference	dictionary		required		See Reference Dictionary.
stall	dictionary		optional		See Stall Dictionary. If not present, no stall model will be added.

Reference Dictionary

The vehicle→properties→components→aerodynamics→reference dictionary contains the following parameters that are used to scale the dimensionless aerodynamic information given.

Key	Value Type	Allowed Values	Required / Optional	Default Value	Description
area[ft^2]	float	> 0	required		Reference area for scaling the aerodynamic properties specified.
longitudinal_length[ft]	float	> 0	required		Reference longitudinal length for scaling the aerodynamic properties specified.
lateral_length[ft]	float	> 0	required		Reference lateral length for scaling the aerodynamic properties specified.

Stall Dictionary

The vehicle→properties→components→aerodynamics→stall dictionary contains the following:

Key	Value Type	Allowed Values	Required / Optional	Default Value	Description
include_stall	boolean	true, false	optional	false	If true, the aerodynamic results will be corrected for stall.
blending_angle[deg]	float	> 0	optional	25.0	Angle at which 50% of the blending has occurred for the stall model.
blending_factor	float	> 0	optional	40.0	Factor used to specify how abrupt the stall is. Factors of 20 to 100 typically produce reasonable results, with 20 for a soft stall and 100 for an abrupt stall.

Generic Model Dictionary

A generic model dictionary can be used for the aerodynamic model, and/or the inertia model of an aircraft. It contains the following:

Key	Value Type	Allowed Values	Required / Optional	Default Value	Description
custom_variables	dictionary		optional		See Custom Variables Dictionary.
equations	dictionary		optional		See Equations Dictionary.
database_directory	string		optional		Path to folder containing the list of databases to be used.
database_list	string array of arbitrary length		optional		Array of database file names to be used. These must be .csv files. See Database Structure.

Custom Variables Dictionary

A model→custom_variables dictionary contains a set of constants or sub-directories, each defining the value of a custom variable. The name of the sub-directory key (variable name) is arbitrary and can be chosen by the user. However, these variables must be defined in alphabetical order and will be computed in alphabetical order. Therefore, no variable in the list can depend on a variable that occurs later in the list. Each custom variable sub-directory can contain either a constant value, or an arbitrary number of entries composed of the following list

of independent variables. Each of the descriptors of these independent variables is relative to the component, not the vehicle.

Aerodynamic Independent Variables:

Keywords	Description
1	Unity.
alpha	Angle of attack in radians.
beta	Sideslip angle in radians.
pbar	Dimensionless rolling rate.
qbar	Dimensionless pitching rate.
rbar	Dimensionless yawing rate.
Any custom variable appearing higher in the list	
Name of any control effector	See Control Effectors Dictionary.

Inertia Independent Variables:

Keywords	Description
1	Unity.
Any custom variable appearing higher in the list	
Name of any control effector	See Control Effectors Dictionary.

Any independent variables that are separated by an underscore will be multiplied together at runtime.

For example, the following is a custom_variables dictionary that could be specified in an aerodynamic model.

```
    "custom_variables" : {
        "CD1" : 0.1,
        "CL1" : {
            "1" : 0.0535,
            "alpha" : 3.84
        },
```

```
        "CS1" : {
            "beta" : -1.098,
            "pbar" : 0.098,
            "CL1_pbar" : 0.002,
            "rbar" : 1.121,
            "aileron" : 0.078,
            "rudder" : 0.168
        }
    }
```

In this example, the user has created three custom variables named CD1, CL1, and CS1. Note that they are listed in alphabetical order. The first variable, CD1, is simply a constant. At runtime, the value of each non-constant custom variable will be computed by multiplying the constant value specified for each parameter by each of the independent variables, parsed by the underscore and summed with all other components in the dictionary. For example, CL1 will take on the value CL1 = 0.0535×1 + 3.84×alpha, where alpha is the component angle of attack in radians. Note that CS1 is a function of CL1. Hence, once the value for CL1 has been computed, the value of CS1 can be found from CS1 = -1.098×beta + 0.098×pbar + 0.002×CL1×pbar + 1.121×rbar + 0.078×aileron + 0.168×rudder. In this example, aileron and rudder are viable independent variables that come from definitions in the Control Effectors Dictionary.

Equations Dictionary

The model→equations dictionary is identical to the Custom Variables Dictionary, with one exception. All sub-directory names in this dictionary must be known keywords of dependent variables. The list of allowed dependent variables depends on what the model is being used for.

For an aerodynamic model, the following dependent variables are allowed:

Cx
Cy
Cz
Cl (small “L” = rolling-moment coefficient)
Cm
Cn
CL (Capital “L” = lift coefficient)
CD
CS (side force coefficient)

For an inertia model, the following dependent variables are allowed:

Ixx[slug-ft^2]
Iyy[slug-ft^2]
Izz[slug-ft^2]
Ixy[slug-ft^2]
Ixz[slug-ft^2]

Iyz[slug-ft^2]

Each of these can be specified as constant or as a function of independent variables, the same way custom variables are defined in the Custom Variables Dictionary.

Database Structure

A generic model dictionary can contain a list of datasets to be used for the model. During program runtime, the properties are interpolated from these tables. Results from each table are summed to give the final properties of the component.

The database information is read in from .csv files with the following format:

1. A pound symbol can be used at the beginning of any line to produce a comment (that line is ignored when read in).
2. Any non-commented lines above the data header can include parameters. Parameters are denoted by any line starting with with a comma-separated word “parameter”. Two parameters are required for the aerodynamic databases.
    a. Number of Independent Variables: This parameter is denoted by the line: “parameter,independent_variables,N” (where N is the number of independent variables in the dataset). This simply tells the interpolation code how to categorize each column of the data. The first N columns are flagged as independent variables, and the remaining columns are flagged as dependent variables.
    b. Coordinate System of the Data: This parameter is denoted by the line: “parameter,coordinate_system,” followed by any of the strings: “body”, “stability”, or “wind”. This tells the code how to rotate the aerodynamic data to get it into the body-fixed coordinate system before application to the equations of motion.
3. The first line that is not commented and is not a parameter is the data header row. This is a list of labels for the columns of data. For example, a header could be: “alpha[deg], Cx, Cy, Cz”.
4. The remaining non-commented rows contain comma-separated values of data.
5. All of the following are used as delimiters when parsing the datafile:
    a. Comma
    b. Semicolon
    c. Tab

It is best to organize the data when you produce the data file. However, the code will sort the data when it is read in. It will sort the data by values (lowest to highest) in order of the columns of the independent variables. Here is an example aerodynamic dataset:

```
# Wind tunnel data taken on <date>
# Any notes you want to add, including reference areas, lengths, etc.
parameter,independent_variables,3
parameter,coordinate_system,body
```

```
elevator[deg],beta[deg],alpha[deg],Cx,Cy,Cz,Cl,Cm,Cn
-25.0,-30.0,-20.0,-0.18370,0.3677,1.19400,-0.00600,0.20590,-0.06330
-25.0,-30.0,-15.0,-0.17140,0.4019,0.99600,-0.00480,0.16980,-0.06210
-25.0,-30.0,-10.0,-0.15310,0.4367,0.79300,-0.00330,0.14260,-0.06780
-25.0,-30.0,-5.0,-0.11510,0.5538,0.41000,0.02980,0.16200,-0.08500
-25.0,-30.0,0.0,-0.09070,0.6218,0.18000,0.02760,0.15300,-0.09950
-25.0,-30.0,5.0,-0.05140,0.6544,-0.09000,0.03900,0.14700,-0.10440
-25.0,-30.0,10.0,-0.00790,0.6255,-0.34000,0.05620,0.15000,-0.09810
-25.0,-30.0,15.0,0.03540,0.5885,-0.61000,0.07370,0.16700,-0.09760
-25.0,-30.0,20.0,0.07400,0.5783,-0.87000,0.07610,0.15100,-0.06770
-25.0,-30.0,25.0,0.10920,0.5005,-1.17000,0.09100,0.12000,-0.04880
-25.0,-30.0,30.0,0.09150,0.3751,-1.31500,0.07430,0.10800,-0.01020
-25.0,-30.0,35.0,0.10790,0.3292,-1.52000,0.07040,0.08200,-0.00280
-25.0,-30.0,40.0,0.13060,0.447,-1.60000,0.06650,0.11300,-0.00370
-25.0,-30.0,45.0,0.15350,0.1634,-1.56000,0.07880,0.09300,-0.01200
-25.0,-30.0,50.0,0.14710,0.1366,-1.30000,0.06050,-0.01500,-0.03730
-25.0,-30.0,55.0,0.15540,0.1735,-1.70500,0.04530,0.01900,-0.04490
-25.0,-30.0,60.0,0.15010,0.2233,-1.70000,0.06100,-0.03600,-0.00550
-25.0,-30.0,70.0,0.15010,0.2609,-1.69000,0.07130,-0.30700,0.02320
-25.0,-30.0,80.0,0.16850,0.3055,-1.93500,0.06140,-0.36500,0.02360
-25.0,-30.0,90.0,0.17120,0.3078,-1.96000,0.06010,-0.52600,0.0319
-25.0,-25.0,-20.0,-0.18530,0.307,1.27200,0.00650,0.19370,-0.06670
-25.0,-25.0,-15.0,-0.17650,0.322,1.05700,0.00590,0.16500,-0.05790
-25.0,-25.0,-10.0,-0.16270,0.3823,0.83200,0.00950,0.15790,-0.05880
-25.0,-25.0,-5.0,-0.12320,0.4778,0.41000,0.02450,0.17700,-0.07610
```

The code looks for “key words” in the data labels. Allowed key words depend on whether it is an aerodynamic model or inertia model.

For an aerodynamic model, it will recognize the following INDEPENDENT variables:

- alpha
- beta
- pbar
- qbar
- rbar
- Any other name of a control effector listed in the Control Effectors Dictionary.

The code will recognize the following DEPENDENT variables:

- Cx
- Cy
- Cz
- Cl (small “L” = rolling-moment coefficient)
- Cm
- Cn
- CL (Capital “L” = lift coefficient)

CD
CS (side force coefficient)

The code recognizes [rad] or [deg] units in square brackets and treats them accordingly. Note that these units should be put in square brackets and part of the label name with no space between the name and the units (i.e. alpha[deg] or beta[deg]).

For an inertia model, it will recognize the following INDEPENDENT variables:
Any name of a control effector listed in the Control Effectors Dictionary.

The code will recognize the following DEPENDENT variables:
Ixx[slug-ft^2]
Iyy[slug-ft^2]
Izz[slug-ft^2]
Ixy[slug-ft^2]
Ixz[slug-ft^2]
Iyz[slug-ft^2]

Control Effectors Dictionary

The vehicle→properties→control effectors dictionary contains a list of dictionaries. These dictionaries must be named with integers in ascending order starting from “0”. Each sub-dictionary contains the following:

Key	Value Type	Allowed Values	Required / Optional	Default Value	Description
name	string	arbitrary	required		Name of the control effector. This name must match any relevant references in the aerodynamics model associated with the parent component of the control effector.
parent_component	string	Any vehicle component	required		Name of the parent component associated with this control effector.
units	string	deg or empty	optional	empty	Units of measurement used for the control effector. Only current options are deg or empty.
magnitude_limits	float array of length 2	arbitrary	required		Minimum and maximum values limiting total value of the control effector in the same units as “units”.
rate_limits[/s]	float array of length 2	arbitrary	required		Minimum and maximum values limiting the actuation rate of the control effector in units of “units” per second.
time_constant[s]	float	> 0	required		First-order lag of control effector.

Initial Dictionary

The vehicle→initial dictionary contains the following:

Key	Value Type	Allowed Values	Required / Optional	Default Value	Description
airspeed[ft/s]	float	>= 0	optional	0.0	Initial airspeed of the vehicle.
mach	float	>= 0	optional	0.0	Initial Mach number of the vehicle. Either airspeed or Mach can be specified.
altitude[ft]	float	arbitrary	optional	0.0	Initial altitude of the vehicle.
latitude[deg]	float	arbitrary	optional	0.0	Initial latitude of the vehicle.
longitude[deg]	float	arbitrary	optional	0.0	Initial longitude of the vehicle.
heading_angle[deg]	float	arbitrary	optional	0.0	Initial heading angle of the vehicle.
type	string	state, trim	required		This specifies how the aircraft will be initialized.
state	dictionary		optional		Required if type = “state”. See State Dictionary.
trim	dictionary		optional		Required if type = “trim”. See Trim Dictionary.

State Dictionary

The vehicle→initial→state dictionary contains the following:

Key	Value Type	Allowed Values	Required / Optional	Default Value	Description
elevation_angle[deg]	float	arbitrary	optional	0.0	Initial elevation angle of the vehicle.
bank_angle[deg]	float	arbitrary	optional	0.0	Initial bank angle of the vehicle.
alpha[deg]	float	arbitrary	optional	0.0	Initial angle of attack of the vehicle.
beta[deg]	float	arbitrary	optional	0.0	Initial sideslip angle of the vehicle.
p[deg/s]	float	arbitrary	optional	0.0	Initial rolling rate of the vehicle.
q[deg/s]	float	arbitrary	optional	0.0	Initial pitching rate of the vehicle.
r[deg/s]	float	arbitrary	optional	0.0	Initial yawing rate of the vehicle.
control_effectors	float array of same length as number of control effectors	arbitrary	optional		Initial control effector values in units specified in Control Effectors Dictionary. The order must match the order the control effectors are defined in the Control Effectors Dictionary. Required if control effectors have been defined.

Trim Dictionary

The vehicle→initial→trim dictionary contains the following:

Key	Value Type	Allowed Values	Required / Optional	Default Value	Description
type	string	sct, shss	required		Specifies the type of trim to be conducted where sct is steady coordinated turn and shss is steady-heading sideslip. SCT requires bank_angle[deg] to be specified. SHSS requires bank_angle[deg] or sideslip_angle[deg] to be specified. Either type requires elevation_angle[deg] or climb_angle[deg] to be specified.
elevation_angle[deg]	float	arbitrary	optional		Initial elevation angle of the vehicle.
climb_angle[deg]	float	arbitrary	optional		Initial climb angle of the vehicle.
bank_angle[deg]	float	arbitrary	optional		Initial bank angle of the vehicle.
sideslip_angle[deg]	float	arbitrary	optional		Initial sideslip angle of the vehicle.
solver	dictionary		required		See Solver Dictionary.

Solver Dictionary

The vehicle→initial→trim→solver dictionary contains the following:

Key	Value Type	Allowed Values	Required / Optional	Default Value	Description
finite_difference_step_size	float	> 0	optional	0.01	The step size to be used in the finite-difference computation of gradients.
relaxation_factor	float	> 0	optional	0.5	The relaxation factor to be used in the Newton's method solver.
tolerance	float	> 0	optional	1.0e-10	The value of the error residual at which the solution will be considered converged.
max_iterations	int	> 0	optional	100	The maximum number of iterations, at which the solver will exit if the solution has not yet converged.
verbose	boolean	true, false	optional	false	Whether verbose information about the solver should be printed to the screen.

D. Analysis Dictionary

The analysis dictionary is an optional dictionary used to perform additional analysis on the aircraft outside of 6-DoF simulation. If present, this dictionary contains the following:

Key	Value Type	Allowed Values	Required / Optional	Default Value	Description
reference	dictionary		required		A dictionary with reference values for the vehicle. See Reference Dictionary.
export_aero_database	dictionary		optional		See Export Aerodynamic Database Dictionary.
export_linear_aero_model	dictionary		optional		See Export Linear Aerodynamic Model Dictionary.
export_linear_aero_derivatives	dictionary		optional		See Export Linear Aerodynamic Derivatives Dictionary
export_linear_state_space_model	dictionary		optional		See Export Linear State-Space Model Dictionary

Export Aerodynamic Database Dictionary

The analysis→export_aero_database dictionary contains an arbitrary number of sub-dictionaries, each with a unique, arbitrary name. The name of each sub-dictionary is appended with ".csv" to create the file name for the corresponding database. Hence, the number of sub-directories corresponds to the number of databases that will be generated. Each sub-dictionary contains the following about a single database to be created:

Key	Value Type	Allowed Values	Required / Optional	Default Value	Description
independent_variables	string	alpha[deg], beta[deg], p[deg/s], q[deg/s], r[deg/s], pbar, qbar, rbar, any control effector display name.	required		A single string with all independent variables to be used in the database creation delimited with an underscore "_". If units pertain to the independent variable, they must be included. Any independent variables not included are set to zero for the database creation.
dependent_variables	string	Cx, Cy, Cz, Cl, Cm, Cn, CL, CD, CS	required		A single string with all dependent variables to be output in the database delimited with an underscore "_".
<independent_variable_name>_range	float array	arbitrary	required		For each independent variable listed in the first parameter, there must be an associated "_range" denoted as an array. If the length of this array is 3, the first two values denote the bounds of the range, and the last value denotes the increment to be used within that range. If the length is anything other than 3, the values in the array are used as a list of values for the independent variable.

Export Linear Aerodynamic Model Dictionary

The analysis→export_linear_aero_model dictionary contains the following:

Key	Value Type	Allowed Values	Required / Optional	Default Value	Description
finite_difference_step_size	float	> 0	optional	0.01	The step size to be used in the finite-difference computation of gradients.

Export Linear Aerodynamic Derivatives Dictionary

The analysis→export_linear_aero_derivatives dictionary contains the following:

Key	Value Type	Allowed Values	Required / Optional	Default Value	Description
finite_difference_step_size	float	> 0	optional	0.01	The step size to be used in the finite-difference computation of gradients.

Export Linear State-Space Model Dictionary

The analysis→export_linear_state_space_model dictionary contains the following:

Key	Value Type	Allowed Values	Required / Optional	Default Value	Description
finite_difference_step_size	float	> 0	optional	0.01	The step size to be used in the finite-difference computation of gradients.

If present, the code will export both an A and B matrix, each in a separate CSV file to represent x_dot = Ax + Bu.

4. Outputs

5. Physics

A. Equations of Motion

MAVIS employs six degree-of-freedom rigid-body equations of motion where the inertial coordinate system is assumed to be a flat earth. This neglects the rotation of the earth as well

as any centripetal forces due to high-speed travel. These effects may be added in the future. The full equations of motion and solution method is outlined by Hunsaker [1].

B. Numerical Methods

MAVIS uses 4th-order Runge-Kutta integration as the core numerical method for integrating the states of the aircraft forward in time. This integration scheme has been shown to be nearly identical in accuracy per computation time step as the commonly-used 4th-order Adams-Bashforth-Moulton method [2–4].

C. Turbulence

Dryden Turbulence Model

The implementation of the Dryden turbulence model follows the development of Beal [5] and is outlined in Hunsaker [1]. Gust or turbulence perturbations are included as perturbations to the aerodynamic properties of the components.

6. Interfacing with MAVIS

By default, MAVIS has the following connections built in. MAVIS looks for the associated keys in the Connections dictionary.

Key	Array Length	Array Contents
send_states	14 + number of control effectors	time[s], u[ft/s], v[ft/s], w[ft/s], p[rad/s], q[rad/s], r[rad/s], x[ft], y[ft], z[ft], e0, ex, ey, ez, actual value of each control effector in units specified in Control Effectors Dictionary
send_imu	17	time[s], a_x[ft/s^2], a_y[ft/s^2], a_z [ft/s^2], p_dot[rad/s^2], q_dot[rad/s^2], r_dot[rad/s^2], p[rad/s], q[rad/s], r[rad/s], e0, ex, ey, ez, phi[rad], theta[rad], psi[rad]
send_pitot	2	time[s], u[ft/s]
send_gps	4	time[s], x[ft], y[ft], z[ft]
send_graphics		Time[s], u[ft/s], v[ft/s], w[ft/s], ax[ft/s^2], ay[ft/s^2], az[ft/s^2], x[ft], y[ft], z[ft], e0, ex, ey, ez, phi[rad], theta[rad], psi[rad], speed of sound[ft/s],ground altitude[ft],gravity[ft/s^2], actual value of each control effector in units specified in Control Effectors Dictionary
receive_controls	number of control effectors	commanded values for each control effector in units specified in Control Effectors Dictionary

Other connections can be easily added by modifying the source code.

7. Examples

A. Vehicles

Sphere

Run the sphere example by navigating to examples/sphere and running

```
mavis sphere.json
```

You can open and plot the results from the output file sphere_states.csv.

Arrow

RQ-21 Fixed-Wing Aircraft (Component Buildup)

Quad-Rotor (Component Buildup)

https://www.dji.com/mavic-3-pro

Transition Vehicle (Component Buildup)

https://www.foxtechfpv.com/foxtech-great-shark-330-pro-vtol.html

F-16 (Custom Aerodynamic Model)

F-16 (Custom Aerodynamic Database)

In 1979, NASA published a dataset based on wind-tunnel data that was used for an F-16 flight simulator. The original publication can be found here. A digitized version of this original dataset can be found here. This original information has been reformatted into a form that can be used in this flight simulator. The datasets in this format can be found here.

B. Controls

PID Controller for Fixed-Wing Aircraft

C. Visualization

Paraview

Unreal Engine

8. Developer Notes

To Do:

- Check aerodynamic coefficients for _body _wind _stability coordinates and how those are implemented
- Write the aerodynamic database documentation
- Write the aerodynamic coefficients documentation
- Modify code so order of control effectors doesn't matter(?)
- Add examples
- Have student check results vs. SAASHA

- Finish turbulence model development
- Not sure angular momentum of rotor is included

To find out how many lines of code are in the project, cd to the directory in question and use the command

```
git ls-files | xargs wc -l
```

I believe in order to do this it requires it to be a git repository.

9. References

[1] Hunsaker, D., *Simulation of Flight*, 2024.

[2] Gerald, C. F., and Wheatley, P. O., *Applied Numerical Analysis*, 6th ed., Addison Wesley Longman, Reading, MA, 1999, pp. 451 – 477.

[3] Hoffman, J. D., Numerical Methods for Engineers and Scientists, McGraw-Hill, New York, 1992, pp. 225 – 266.

[4] Phillips, W. F., Hailey, C. E., and Gebert, G. A., “Review of Attitude Representations Used for Aircraft Kinematics,” *Journal of Aircraft*, Vol. 38, No. 4, 2001.

[5] Beal, T. R., “Digital Simulation of Atmospheric Turbulence fro Dryden and von Karman Models,” *Journal of Guidance, Control, and Dynamics*, Vol. 16, No. 1, 1993.