openRocket_flight Documentation

UTAT Rocketry

June 2021

1 Inputs

1.1 Files

To successfully run the simulation, two files are needed: the main open rocket file ending with `.ork` and the respective engine file ending with `.rse`. By default, both files should be placed in a folder named `Simulation`, which is in the same directory as the code.

The file structure should be as follows:

```
Project Folder
 |
 +-- openRocket_flight.py
 |
 +-- OpenRocket-15.03.jar
 |
 +-- Simulation
 |   |
 |   +-- <ROCKET>.ork
 |   |
 |   \-- <MOTOR>.rse
 |
 \-- <YOUR CODE>.py
```

1.2 Class Constructor

```
openRocket_flight(simulation, engine=None, folder='Simulation')
```

Parameters

- simulation : str
  The name of the rocket file, with the extension.

- engine : str
  The name of the motor file, with the extension. By default, there is no custom motor attached.

- folder : str
  Name of the folder where both simulation files are stored. The default location is in the Simulation folder.

2 Output

> **Warning**: Remember to call the function `run()` before retriving the outputs.

State Vector It can be accessed via `flight_run.state_vector`. The vector is a numpy array in this form:

$$\begin{bmatrix} t \\ s_x \\ s_y \\ s_z \\ v_x \\ v_y \\ v_z \\ a_x \\ a_y \\ a_z \\ q_w \\ q_x \\ q_y \\ q_z \\ \omega_{x_1} \\ \omega_{x_2} \\ \omega_{x_3} \\ \alpha_{x_1} \\ \alpha_{x_2} \\ \alpha_{x_3} \end{bmatrix}$$

Note, in this vector, x, y, z are the principle axis of the launch site, x being the direction parallel to the wind [Source]; while x_1, x_2, x_3 are the principle axis of the rocket, which changes as the rocket rotates.

Each variable in the vector is a numpy array containing data collected through the simulation.

In addition, `flight_run.euler_axis` and `flight_run.quaternion` are also available as the stand-alone vectors for convenience.

Parachute Deploy The exact second of the first parachute being deployed after launch can be accessed by `flight_run.recovery_time`.

Raw Data All the raw data are organized into their respective family shown in Table 1. The data can be accessed by `flight_run.<FAMILY>.<TYPE>`

Table 1: List of all raw data available.

Family	**Type**
aerodynamic_coefficients	TYPE_FRICTION_DRAG_COEFF
	TYPE_PRESSURE_DRAG_COEFF
	TYPE_BASE_DRAG_COEFF
	TYPE_NORMAL_FORCE_COEFF
	TYPE_PITCH_MOMENT_COEFF
	TYPE_YAW_MOMENT_COEFF
	TYPE_SIDE_FORCE_COEFF
	TYPE_ROLL_MOMENT_COEFF
	TYPE_ROLL_FORCING_COEFF
	TYPE_ROLL_DAMPING_COEFF
	TYPE_PITCH_DAMPING_MOMENT_COEFF
	TYPE_YAW_DAMPING_MOMENT_COEFF
	TYPE_DRAG_COEFF
	TYPE_AXIAL_DRAG_COEFF
atmospheric_conditions	TYPE_WIND_VELOCITY
	TYPE_AIR_TEMPERATURE
	TYPE_AIR_PRESSURE
	TYPE_SPEED_OF_SOUND
forces	TYPE_GRAVITY
	TYPE_THRUST_FORCE
	TYPE_DRAG_FORCE
kinematics_dynamics	TYPE_AOA
	TYPE_ROLL_RATE
	TYPE_PITCH_RATE
	TYPE_YAW_RATE
	TYPE_MACH_NUMBER
	TYPE_REYNOLDS_NUMBER
	TYPE_CORIOLIS_ACCELERATION
	TYPE_POSITION_X
	TYPE_POSITION_Y
	TYPE_POSITION_XY
	TYPE_POSITION_DIRECTION
	TYPE_VELOCITY_XY
	TYPE_ACCELERATION_XY
	TYPE_LATITUDE
	TYPE_LONGITUDE

	TYPE_ORIENTATION_THETA TYPE_ORIENTATION_PHI TYPE_VELOCITY_TOTAL TYPE_ACCELERATION_TOTAL TYPE_ALTITUDE TYPE_VELOCITY_Z TYPE_ACCELERATION_Z
rocket_properties	TYPE_REFERENCE_LENGTH TYPE_REFERENCE_AREA TYPE_MASS TYPE_MOTOR_MASS TYPE_LONGITUDINAL_INERTIA TYPE_ROTATIONAL_INERTIA TYPE_CP_LOCATION TYPE_CG_LOCATION TYPE_STABILITY TYPE_PROPELLANT_MASS
simulation_information	TYPE_TIME_STEP TYPE_COMPUTATION_TIME TYPE_TIME