MachUp Input Files

23 February 2016

In order to run cases with Machup, aircraft parameters must be specified in an input file. This file is included for editing under the name `input.json`. This document gives instructions on how to customize and run your own input file in MachUp.
First we open the file:

- Find the `input.json` file in the `bin` directory. This directory was specified when MachUp was first compiled.
- Open the `input.json` file to begin editing.
- Edit the `input.json` file according to the instructions given below.
- To run the input file, make sure that the `input.json` file is located in the same directory as the `MachUp.out` file. from the terminal, change your directory to the directory in which `MachUp.out` is located, and then enter: `./MachUp.out input.json`

The `input.json` file contains the aircraft parameters that MachUp needs in order to give resluts for any aircraft. It allows the user to define aircraft geometry, select the solver type, and specify the type and format for the results. The commands in the `input.json` file are explained below.

Routines

- **airfoil_DB:** Looks in the specified directory to find airfoil information if custom airfoil parameters are not defined below. The airfoil coordinates are needed in order to build a `.stl` file, and airfoil properties can be saved in a `.json` file to avoid the necessity of inputing properties each time common airfoils are used. In order to specify the directory in which these airfoil parameters are stored, type "*path_to_file*". If the airfoil information is stored in the same directory as the input file, type ".".
- **run:** The program performs the subroutines listed below. If it is desired to exclude any of these subroutines, simply type an x in front of the subroutine name, and it will not be performed. (*ex. "distributions"* becomes *"xdistributions"*)
  - **stl:** Imports the wing geometry to the `input_view.stl` file, which is saved in the same directory as the input file. This file can be imported to CAD or CFD software for further analysis. (Note: In order to create an .stl file, a `.txt` file containing the airfoil coordinates must be included in the directory specified in airfoil_DB.)
  - **plot:** Plots the wing in plotmtv and writes the wing geometry to the `input_PlotData.txt` file which can be run with plotmtv to later visualize the wing. In order for the wing to plot automatically when the plot subroutine is run, plotmtv must be included in your path.
  - **forces:** Calculates inviscid, viscous, and total force coefficients on each wing and on the entire aircraft and saves the results in the `input_forces.json` file.
  - **distributions:** Calculates the force distributions on the wing and writes the results to `input_distributions.json` file unless specified otherwise by the user[a]. (Note: The distributions routine will also write the forces and moments to the `input_forces.json` file.)

- **derivatives:** Calculates the stability, control, and damping derivatives, along with the static margin for the wing and total aircraft. The results are written to the `input_derivatives.json` file unless specified otherwise by the user[a]. (Note: any user-specified file for derivatives must be a .json file.)
- **stallonset:** Begins at the user specified angle of attack and steps through angles of attack until the onset of stall at any section along the wing. The user can specify the starting angle of attack by typing in `"start_alpha":` *insert_value*. If no value is specified by the user, the program will default to an angle of attack of 0.0. The angle of attack at which stall onset occurs, along with the spanwise location of the stall onset is written to the `input_stallonset.json` file unless specified otherwise by the user[a].
- **aerocenter:** Calculates the location of and moment about the aerodynamic center of the aircraft. The results are written to the `input_aerocenter.json` file unless specified otherwise by the user[a].
- **pointloads:** Calculates the loads at the control points of the wing. The results are written to the `input_pointloads.txt` file unless specified otherwise by the user[a].
- **targetcl:** Calculates the angle of attack required to achieve a user specified target lift coefficient. The following customizations can be made to this routine:
  - **CL:** Change the desired target CL by typing `"CL":` *insert_value*.
  - **delta:** Change the desired angle step size by typing `"delta":` *insert_value*. (Default: 0.5)
  - **convergence:** Change the desired convergence criterion by typing `"convergence":` *insert_value*. (Default: 1.0e-10)
  - **relaxation:** Change the desired relaxation factor by typing `"relaxation":` *insert_value*. (Default: 1.0)
  - **maxiter:** Change the desired maximum number of iterations by typing `"maxiter":` *insert_value*. (Default: 50)

  If any of these values are left unspecified, the program will assign the default values listed. The results will be written to the `input_targetCL.json` file unless specified otherwise by the user[a].
- **spanloads:** Calculates the loads at spanwise locations along the wing. The results are written to the `input_pointloads.txt` file unless specified otherwise by the user[a].
- **pitchtrim:** Trims the aircraft using the control surface specified and calculates values for angle of attack, control surface deflection, lift coefficient, and moment coefficient. The following customizations can be made to this routine:
  - **control:** Specify the control surface by typing `"control":` *"insert_surface"*. (Default: Elevator)
  - **CL:** Specify the lift coefficient by typing `"CL":` *insert_value*
  - **Cm:** Specify the moment coefficient by typing `"Cm":` *insert_value* (Default: 0.0)
  - **delta:** Change the desired step size by typing `"delta":` *insert_value*. (Default: 0.5)
  - **convergence:** Change the desired convergence criterion by typing `"convergence":` *insert_value*. (Default: 1.0e-10)
  - **relaxation:** Change the desired relaxation factor by typing `"relaxation":` *insert_value*. (Default: 1.0)
  - **maxiter:** Change the desired maximum number of iterations by typing `"maxiter":` *insert_value*. (Default: 50)

  If any of these values are left unspecified, the program will assign the default values listed. The results will be written to the `input_pitchtrim.json` file unless specified otherwise by the user[a].
- **report:** Allows the user to select specific data from any file returned from any of the above routines and write it to a report file. This is done by typing *"name_of_data"*:{`"name":` *"name_of_report"*, `"file":` *"name_of_outputfile.file_type"*} The report will be saved under the name specified in the report command.

[a]**The user can specify the file name to which values returned by the routines are written by typing** `"filename":` *"insert_name"* **and the file type by typing** `"output":` *"insert_type"* **or by simply typing** `"filename":` *"insert_name.file_type"*

Solver

- **solver:** The solver used to perform the subroutines listed above. The user can either specify linear or nonlinear solvers and customize the convergence criteria and relaxation factors.
  - **type:** Change the solver type by either specifying `"linear"` or `"nonlinear."`
  - **convergence:** Customize the convergence criterion by changing `"convergence"` to a user-specified value. (Default: 1.0e-6)
  - **relaxation:** Customize the relaxation factor by changing `"relaxation"` to a user-specified value. (Default: 0.9)
  - **maxiter:** Customize the max number of iterations by changing `"maxiter"` to a user-specified value. (Default: 100)

Aircraft Geometry

- Plane
  - **Name:** Choose a name for your airplane. Unless otherwise specified, the name will default to `"myairplane"`.
  - **CGx:** Specify the x location of the center of gravity of the aircraft.
  - **CGy:** Specify the y location of the center of gravity of the aircraft.
  - **CGz:** Specify the z location of the center of gravity of the aircraft.

  Note: The coordinate system is defined based on a standard flight mechanics coordinate system with the positive x direction from the CG forward toward the nose of the aircraft, the positive y direction from the CG out toward the right side of the aircraft, and the positive z direction from the CG in the vertical downward direction, as shown in figure 1 below.

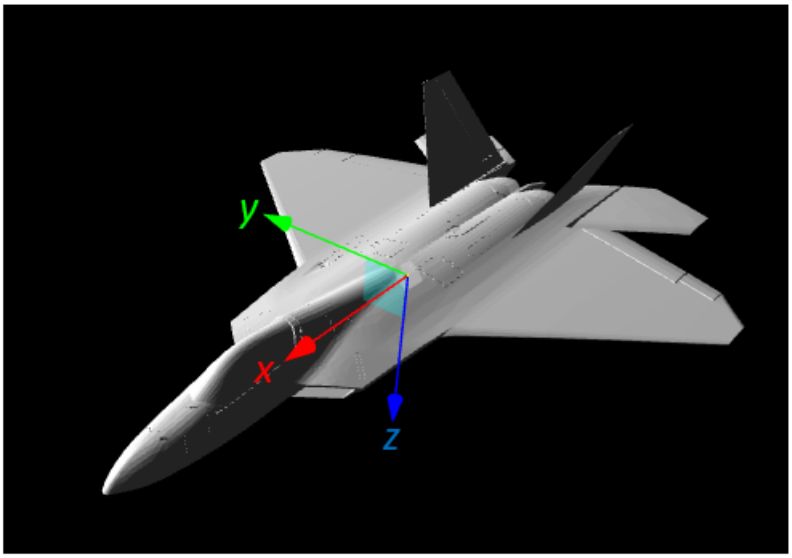

Figure 1: Flight Dynamics Coordinate System

- **reference:** Define reference areas used to non-dimensionalize forces.
  - **area:** Specify the reference area. The reference area generally used is the wing planform area.

  - **longitudinal_length:** Specify the longitudinal reference length. The longitudinal reference length generally used is the mean chord length.
  - **lateral_length:** Specify the lateral reference length. The lateral reference length generaly used is the wingspan.

- **condition:** Define flight conditions related to freestream velocity and ground proximity.
  - **alpha:** Specify the angle of attack, in degrees.
  - **beta:** Specify the sideslip angle, in degrees. (Default: 0.0)
  - **ground:** Specify the aircraft distance from the ground. The ground proximity will be used to calculate ground effect. (Default 0.0)
  - **omega:** Specify the rates of rotation for roll, pitch, and yaw.
    - **roll:** Rate of rotation for roll, in radians^{-1}. (Default 0.0)
    - **pitch:** Rate of rotation for roll, in radians^{-1}. (Default 0.0)
    - **yaw:** Rate of rotation for roll, in radians^{-1}. (Default 0.0)

- **controls:** Define the control surface geometries for the aircraft.
  - **aileron:** Aileron properties.
    - **is_symmetric**: Enter `1` for symmetric aileron, and `0` for non-symmetric aileron.
    - **deflection:** Specify deflection angle, in degrees. (Note: True control surface deflection is the value specified here, multiplied by the value given in the `mix`) field.
  - **elevator:** Elevator Properties
    - **is_symmetric:** Enter `1` for symmetric elevator, and `0` for non-symmetric elevator.
    - **deflection:** Specify deflection angle, in degrees. (Note: True control surface deflection is the value specified here, multiplied by the value given in the `mix`) field.

- **wings:** Wing Properties. Type `"wing_1"` for the first wing, `"wing_2"` for the second wing, etc. These properties will refer to the wing under which they are specified. To add a second wing, simply reproduce this section with the second wing properties after completing it with the first wing properties. (Note: the properties for each wing should be included in brackets following that wing name. (ex. `"wing_1":{`*properties*`}`))
  - **ID:** Define an ID number for this wing.
  - **side:** Enter `"right"` for a wing on the right side only, `"left"` for a wing on the left side only, and `"both"` for a wing on both sides.
  - **connect:** Define how the wing is connected to other aircraft components.
    - **ID:** Specify the ID number for the part to which this wing is connected. You can only connect to a wing with a lower ID than the current wing. (enter `0` if not connected to any wing)
    - **location:** Define what part of this wing is connected to the base part. The user can specify `"root"` or `"tip"`
    - **dx:** x location of the wing with respect the center of gravity.
    - **dy:** y location of the wing with respect the center of gravity.
    - **dz:** z location of the wing with respect the center of gravity.
    - **yoffset:** y distance between the roots of each semispan. For continuous wing, set `"yoffset":0`.
  - **span:** Wingspan of the given wing.
  - **sweep:** Sweep angle of the wing quarter chord, in degrees.
  - **dihedral:** Dihedral angle of the wing, in degrees.
  - **mounting_angle:** Mounting angle of the wing, in degrees.
  - **washout:** Washout Angle, in degrees.

- **root_chord:** Chord length at the wing root.
- **tip_chord:** Chord length of the wing tip.
- **root_airfoil:** Define the root airfoil.
  - **name:** Enter the name of a file in the directory specified in `airfoil_db` which contains the airfoil properties. This file must be a `.json` file and contain values for the properties below.
  - **properties:** Airfoil Properties.
    - **type:** Specify either `"linear"` or `"datafile"`. The `linear` specification will assume linear lift properties over all ranges of angle of attack based on the properties defined in this section. The `datafile` option will read properties from a file containing airfoil data obtained from an outside source. In order to use the `datafile` option, the user must choose a file from which to read the airfoil properties. This is done by typing `"filename":`*`file_name`*.
    - **alpha_L0:** The zero lift angle of attack for the airfoil.
    - **CL_alpha:** The lift slope of the airfoil.
    - **Cm_L0:** The zero lift moment coefficient of the airfoil about the aerodynamic center.
    - **Cm_alpha:** The moment slope of the airfoil about the aerodynamic center.
    - **CD_min:** The Minimum drag coefficient of the airfoil.
    - **CL_max:** The maximum lift coefficient for the airfoil.
- **tip_airfoil:** Define the tip airfoil.
  - **name:** Enter the name of a file in the directory specified in `airfoil_db` which contains the airfoil properties. This file must be a `.json` file and contain values for the properties below.
  - **properties:** Airfoil Properties.
    - **type:** Specify either `"linear"` or `"datafile"`. The `linear` specification will assume linear lift properties over all ranges of angle of attack based on the properties defined in this section. The `datafile` option will read properties from a file containing airfoil data obtained from an outside source. In order to use the `datafile` option, the user must choose a file from which to read the airfoil properties. This is done by typing `"filename":`*`file_name`*.
    - **alpha_L0:** The zero lift angle of attack for the airfoil.
    - **CL_alpha:** The lift slope of the airfoil.
    - **Cm_L0:** The zero lift moment coefficient of the airfoil about the aerodynamic center.
    - **Cm_alpha:** The moment slope of the airfoil about the aerodynamic center.
    - **CD_min:** The Minimum drag coefficient of the airfoil.
    - **CL_max:** The maximum lift coefficient for the airfoil.
- **grid:** The number of spanwise segments into which the wing is divided for numerical analysis.
- **control:** Define the Control Surface properties.
  - **span_root:** The spanwise location of the control surface root, normalized by wingspan.
  - **span_tip:** The spanwise location of the control surface tip, normalized by wingspan.
  - **chord_root:** Control surface width at the control surface root, normalized by the chord length.
  - **chord_tip:** Control surface width at the control surface tip, normalized by the chord length.
  - **is_sealed:** Enter `1` for sealed control surface, and `0` for non sealed control surface.
  - **mix:** Defines the ratio of deflection for the control surfaces.
    - **aileron:** Percent aileron deflection with respect to deflection specified in `controls`.
    - **elevator:** Percent elevator deflection with respect to deflection specified in `controls`.