Guide for Folder `1 - Grid & CFD` inputs.

December 24, 2021

1 Introduction

This PDF reports firstly how to generate a single grid file and perform a single CFD simulation within Folder `1 - Grid & CFD`; furthermore, a list about the items, the available options and the respective meanings within the input files are presented.
Among the various Python scripts inside the cited Folder, only those named **main_** are the ones which the user has to modify to insert his/her inputs.
If further information or clarifications are needed, please refer to Folder `Test cases examples` where various test cases are presented.

2 How to run

Open a terminal window and place to the directory `1 - Grid & CFD`.
Then, write the following Python command: See the following command :

```
$ python3 launch_singlerun.py
```

The code will automatically generate the grid files with Gmsh syntax.
If the user has requested the subsequent RANS simulation through the relative input in Python script "main_geo_grid.py", the code requires a check to the grid file through a visualization before continuing. If the grid satisfies user needs, press ENTER to continue and start RANS simulation.

3 Input Files

3.1 main_geo_grid.py

This script contains the most important input parameters regarding geometry and grid details. The following list is subdivided in two sections to distinguish the parameters depending on their relative importance: **MAIN PARAMETERS** and **ADVANCED PARAMETERS**.

MAIN PARAMETERS

- `CFD_run`: defines whether user wants to perform a complete CFD simulation or limiting to grid generation only.

  Available options:

  - *'YES'*: grid generation plus CFD simulation. It will consequently involve the other input script file of the present Folder, that is **main_configuration_file.py**.
  - *'NO'*: grid generation only.

- `typ`: defines the parameterization of each element within airfoil's configuration. It is a list of strings, where each element corresponds to the parametrisation chosen for that particular configuration's element.

  Available options:

  - *"IGP"*: Improved Geometric Parameter (IGP) airfoil parametrisation. It will require 8 parameters through the following item `params` for that element in airfoil's configuration to define the geometry shape.
  - *"NACA4"*: NACA 4-digits airfoil parameterisation. It will require 3 parameters.
  - *"NACA5"*: NACA 5-digits airfoil parameterisation. It will require 4 parameters.
  - *"NACA4_MOD"*: modified NACA 4-digits airfoil parameterisation. It will require 5 parameters.
  - *"NACA16"*: NACA 16-series airfoil parameterisation. It will require 3 parameters. In particular, NACA 16-series is a special case of modified NACA 4-digits airfoil with a leading edge radius index of (4) and maximum thickness located at $x/c = 5$ with c chord.
    Consequently, as example NACA 16-012 is equivalent to an NACA 0012-45, written respectively as 16-series and modified 4-digits.
  - *"MY_FILE_i"*: the chosen element comes from an external file where its (x,y) coordinates are reported. The file for the i-th configuration element must be inserted within Folder `1 - Grid & CFD` and it must be named as "MY_FILE_*i*.dat" where i is an integer value which corresponds to the element position in the airfoil configuration (and, consequently, in the list of strings `typ`).
    The (x,y) coordinates have to be listed as two columns of float values within the .dat file.

- `params`: specifies shape parameters of each element in airfoil's configuration; it is a list of integer values: each term length and value depends on the parametrisation selected in item `typ`.

  Examples:

  - (For IGP): 8 parameters. Example: [0.4, 0.6, -0.1, -0.1, 0.3, 0.15, 0.3, 2.0].
  - (For NACA 4-digit): 3 parameters. Example: params = [0, 0, 12]
  - (For NACA 5-digit): 4 parameters. Example: params = [2, 3, 1, 12]
  - (For NACA 4-digit modified): 5 parameters. Example: params = [0, 0, 20, 6, 2]
  - (For NACA 16 series): 3 parameters. Example: params = [16, 0, 12]
  - (For MY_FILE_i): write directly the string 'MY_FILE_*i*' related to the i-th airfoil.

  Whenever the user is treating multi-element configurations, `params` has to be a list of params, like the following example for a double-element configuration with NACA 0012 as first element and an external file coordinates for the second one:
  params = [[0, 0, 12], ['MY_FILE_2']]

- `TE`: specifies trailing edge type through a list of strings characters.

  Available Options:

  - *'Y'* : cut TE
  - *'N'* : Finite Angle TE

  Note 1): if IGP element has been defined as geometry for an element within the configuration, the trailing edge will be imposed automatically as finite angle, by definition.
  Note 2): if an external geometry is provided for an element within the configuration, the user has to properly define what is the trailing edge shape through the list element within `TE`.

- `alpha`: specifies the absolute angle of attack of each airfoil [deg] through a list of float values.
- `dist`: defines the relative gap between TE and LE of each airfoil w.r.t. previous. For each slot, two float values are required as (x, y) coordinates.
  Example for a double-element airfoil: dist = [[0, 0], [0.025, 0.025]]
  Note: first list term must always be [0, 0].
- `crel`: specifies the relative respective chord of each airfoil [%] through a list of float values.
  Example for a double-element airfoil: dist = [1, 0.3].
  Note: first list term must always be unit value.
- `farfield_size`: represents the farfield boundary distance from the configuration as times unit chord.
- `nodes`: defines a proportional and generic refinement of the grid in the domain. Consequently, each feature of the grid (airfoil's wall surfaces nodes, ellipse nodes etc.) sees its nodes scaled by this factor.
  Reference value: 1.
- `ellipse_dimension`: ellipse dimension, used for grid refinement. Limit values: [0.1, 0.95].
  Example: ellipse_dimension = 0.85
- `ellipse_refining`: ellipse refinement defined by a factor which proportionally scales the total amount of nodes on its perimeter.
- `flow`: defines the typology of flow.
  Available options:
  - *'VISCOUS'*: flow where viscosity effects are not negligible; consequently, a structured region close to configuration surfaces will be automatically imposed.
  - *'EULER'*: flow where viscosity effects are negligible. Consequently, all the following data regarding structured region definition (wall spacing, structured region thickness and progression, wall refining, free-stream flow conditions) are ignored.
- `y_plus`: defines the nominal dimensionless wall distance based on Schlichting flat plate analogy.
  Consequently, it must be positive lower than 1; since the flow behaviour varies with respect to x-coordinate and the analogy with flat plate empirical estimation involves an error due to curvature of airfoils, the nominal value may be imposed with lower values (ex: 0.25), depending also on flow conditions and angle of attack.
- `thick`: structured region thickness. Limit values: [0, 2].
  Note: upper limit has been imposed to avoid distortions of structured elements due to curvatures. It is calculated as [%] of the turbulent boundary layer prediction proposed by Schlichting.
- `progr`: progression of structured elements in normal direction with respect to airfoils' surface.
  Limit values: [1, 1.3].
- `wall_refining`: list of float values which defines the refinement on airfoil configuration's elements.
  Consequently, each term will proportionally scale the total amount of nodes along that configuration's element surface. Suggested value: $1.0 \div 1.3$.
- `Mach`: free-stream Mach number [-].
- `Re`: free-stream Reynolds number [-].
- `mu`: free-stream dynamic viscosity [Pa * s].
- `temp`: free-stream temperature [K].

ADVANCED PARAMETERS

- `ref_airfoil`: in case of multi-element configuration, define which is the airfoil to take as main reference for the angle of attack, relative chord and grid features.
  Inputs could be an integer value or the string ”DEFAULT” takes the airfoil with higher chord as reference.
- `wake_length`: refinement regarding configuration's wake. It is a float value which indicates the length as proportional to ellipse major axis length.
- `wake_progr`: wake refinement is defined by a sequence of points along two straight lines, which angular coefficient is determined by the input angle of attack. Each point position along these straight lines is determined by a geometrical series with the input value of ”wake_progr” as geometrical progression.
- `Mesh`: defines the algorithm selected for unstructured region grid generation. It can be an integer value from 1 to 9 (please refer to Gmsh manual for details). It is also possible to insert the string 'DEFAULT', which corresponds to algorithm number 5 (Delaunay).
- `external_pts`: string term to add (or not) grid points far from the airfoil configuration. These points can help meshing algorithm (strictly depending on geometry and size) and favour general refinement upward the configuration.

  Available options:

  - *'YES'*: adds external points;
  - *'NO'*: does not add external points. Consequently, the following items semicircle_dimension and semicircle_elem_factor will be ignored.
- `semicircle_dimension`: defines external grid semi-circle dimension (radius).
  It takes an integer value, which proportionally scales the semi-circle dimension with respect to grid ellipse's major semi-axis.
  Suggested value: $15 \div 25$.
- `semicircle_dimension`: defines elements' sizes of the points belonged to the external grid semi-circle.
  It takes an integer value, which proportionally scales the smaller grid dimension.
  Suggested value: $200 \div 300$.
- `n`: integer value which defines the number of points for geometry shapes definitions with default parametrisations (IGP, NACA 4-digits, etc.).

4 main_configuration_file.py

This input script defines the CFD features and details read by SU2 to perform the simulations. Compared to the amount of possible inputs which SU2 is capable to accept, the script allows the user to modify the most important CFD details.
Consequently, further details about the available inputs within the script are directly left to SU2 website and literature, where each input is deeply explained.
At the current stage of the tool, note that input variable within this Python script must not be deleted or erased.