Test Matrix Generator User Guide
Version 1.1

Table of Contents

Introduction

Test Matrix Generator is a Python program used to generate generic test matrices for testing. It was designed around the Tkinter (GUI toolkit) and Pandas (data analysis and manipulation module). The main test matrix is stored as a list of Pandas DataFrames where each DataFrame corresponds to a test point group with the columns defined by the test parameters. The TestMatrix class houses the test matrix as well as all functions to manipulate it. The program also has a specific file structure that needs to be maintained which consists of four folders (GUI, Core, Templates, OutputScripts) and one launch script (Launch_GUI.py). The GUI folder houses all the of the GUI elements, the Core folder houses all of the scripts used to interact with the test matrix, the Templates folder houses the user defined parameters and definitions templates, and the OutputScripts folder houses the scripts used to generate the final user friendly test matrix.

Requirements and Usage

This program requires Python 3.6 or greater, Pandas, and Pyyaml.

Python: https://www.python.org

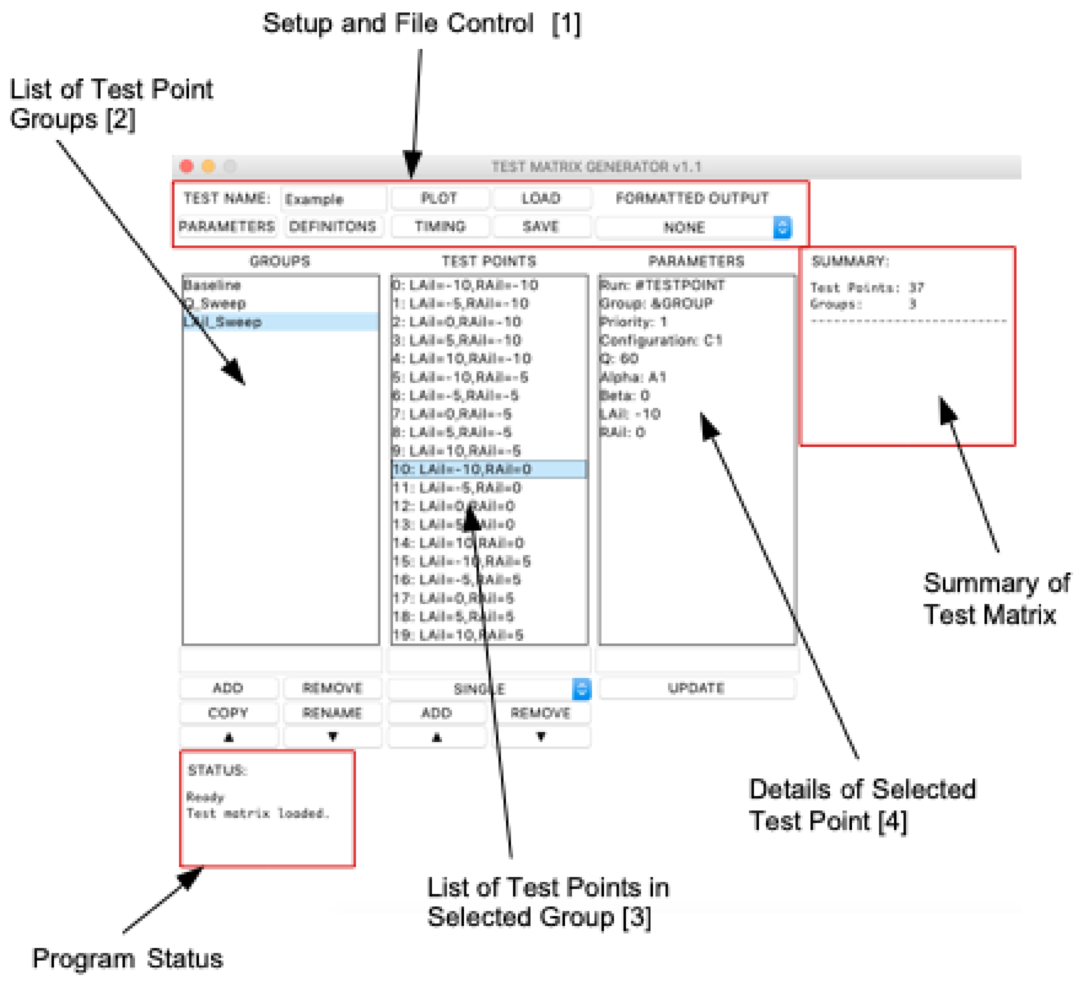
Pandas: 'pip install pandas'

Pyyaml: 'pip install pyyaml'

To run the program, open a new Command Prompt session, navigate to the folder containing Launch_GUI.py and execute the following command: python Launch_GUI.py

Main GUI

The main GUI window is shown in Figure 1. It is divided up into four main sections: setup and file control, test point visualization list boxes (3), summary box, and status box. Details about the setup and file control and test point visualization boxes are given in the following sections. The status box provides feedback to the user when interacting with the GUI. The summary box provides a general overview of the test matrix with the first two lines, test points and groups, always shown. These show the total number of test points and the total number of test point groups. Below these, the information shown will be based on the flags set in the parameters window which is detailed in a later section.



```
TEST NAME: Example
FORMATTED OUTPUT: NONE

GROUPS
Baseline
Q_Sweep
LAil_Sweep

TEST POINTS
0: LAil=-10,RAil=-10
1: LAil=-5,RAil=-10
2: LAil=0,RAil=-10
3: LAil=5,RAil=-10
4: LAil=10,RAil=-10
5: LAil=-10,RAil=-5
6: LAil=-5,RAil=-5
7: LAil=0,RAil=-5
8: LAil=5,RAil=-5
9: LAil=10,RAil=-5
10: LAil=-10,RAil=0
11: LAil=-5,RAil=0
12: LAil=0,RAil=0
13: LAil=5,RAil=0
14: LAil=10,RAil=0
15: LAil=-10,RAil=5
16: LAil=-5,RAil=5
17: LAil=0,RAil=5
18: LAil=5,RAil=5
19: LAil=10,RAil=5

PARAMETERS
Run: #TESTPOINT
Group: &GROUP
Priority: 1
Configuration: C1
Q: 60
Alpha: A1
Beta: 0
LAil: -10
RAil: 0

SUMMARY:
Test Points: 37
Groups:      3
------------------------------

STATUS:
Ready
Test matrix loaded.
```

Figure 1: Main GUI Window

Setup and File Control [1]

This section is used to setup or modify an existing test matrix as well as save it. The test name field is used as the filename or is populated with the file name if an existing matrix is loaded. The Parameters button opens the Parameters window and is used to setup and modify test parameters. The Definitions button opens the Definitions window and is used to manage and define definitions used as parameter values. The Load button is used to load a pre-existing test matrix. This matrix must follow a specific format (.tm). The Save button saves the current test matrix using the .tm format and in the specific format from the Formatted Output menu.

Parameter Window

The parameter window controls the parameters used to define the test. They are the 'columns' of the test matrix. There are two main elements to the parameter window, the list of parameters with their default values (left list box), and the parameter flags (right list box). The user can add, delete, change, or move around the parameters within the list box using the buttons below the box. The order of the parameters in the list box is the order in which the parameters are listed the output test matrix going from left to right. The user may add parameters manually or load them from an existing template. The templates use the YAML standard and are formatted as such:

```
ParameterName:
  VALUE: ParameterValue
  FLAG: ParameterValue
  TIMING:
    VALUE: TimingValue
    FLAG: TimingFlag
```

The timing fields are covered in the Timing Window section. New parameters are added using the entry field directly underneath the list box and are added using the following syntax, NAME:VALUE. If a parameter is to be modified, select the parameter and enter a new value in the entry field. This value can be a number, character, string, or list. There are some predefined strings that can be used to define indexing/naming. These strings will be replaced during the final output and are defined in Table 1.

Table 1: Predefined Parameter Values

#GROUP	Group index number
&GROUP	Group name
#TESTPOINT	Test point index number within the test point group
#IDX	Overall index number of test point across all groups

These can be used in combination, such as #GROUP - #TESTPOINT. If a parameter value is modified and a test matrix already exists, every test point that still has the original default value of the parameter will be updated to the new default parameter. Parameter flags are modified using the same syntax as the modifying the parameter value. The flags are used to determine how the parameters will be handled during the final output and in the summary box on the main GUI and are defined in Table 2.

Table 2: Parameter Flags

-	Null flag Default flag when a new parameter is added.
*	Ignore flag Used to prevent the column from being written in the final output matrix.
S	Summary flag Includes the column in the summary box on the main GUI.
T	Timing flag Indicates that the parameter will be used to estimate test time. Values and timing type are controlled in the Timing Window.

Multiple flags are allowed for a single parameter except when using the null flag. Once all changes have been made in the parameters window, use the Save and Close button to save any changes and close the window. Using the 'x' to close the window will erases any changes.

Definition Window

The definitions window is used to define abbreviations and or symbols used throughout the test matrix. Definitions are optional. For example, for an alpha sweep in a wind tunnel test to user can set the 'Alpha' parameter equal to 'A1' instead of the list of alphas [-2,0,2,...]. This makes the output formatting look better and allows the user to quickly change what values are actually included in 'A1'. The definitions window follows the same input format as the parameters window, NAME:VALUE. In this window the order of the items in the list do not matter. The flags used by the definitions window are defined in Table 3.

Table 3: Definition Flags

-	Null flag Default flag when a new definition is added. During final output, the definition name will be show up in the final matrix.
R	Replace flag The value of the definition will replace the definition in the final output.

Timing Window

This feature is not yet complete.

Plotting Window

The plotting window is used to check the parameter space when designing test points for model generation. The plot window can only be opened it at least one test point group exists. The list boxes are populated using the test point parameters that contain only numerical values, either single values or lists, across all groups. Definitions are checked such that if the user set 'Alpha:A1' then the plot window will include Alpha, provided that A1 is a list or value. Selection of an X and Y variable is required to plot, but selecting a Group and Z parameter are optional. If no Z parameter is selected then zero is used as the Z value for the X,Y plot and if no Group is selected, all groups are used. A convex hull can be added to the data plot using the check box above the plot button. This will show the dividing lines between interpolation regions and extrapolation regions when used to create models.

Save

Any time the save button is pressed, the current test matrix is saved using the internal format as well as the format specified by the Formatted Output menu. These formats are designed to output user friendly matrices to be used during testing. The user may add their own custom output format by placing their output script in the 'OutputScripts' folder. The program will automatically search this folder and add any '.py' scripts to the drop down menu. The script must be a python definition with the definition name and file name being the same.

```
def ScriptName(TestName,TestMatrix):
    # Code here ...
```

TestName is the name of the test (string) and is pulled from the entry field in the main GUI. TestMatrix is the test matrix class and contains all the information contained in the test matrix. Additional details about the TestMatrix class can be found in the Appendix.

Groups [2]

The Groups list box in the main GUI lists all grouped test points by name. A group can have any name as long as the name hasn't already been used. Each group can hold as manny test points as needed and renamed and reordered as needed. The buttons below the list box are used to add, remove, reorder, rename, and copy the group.

Test Points [3]

All test point in the currently selected group are shown here. The name shown is dynamic and is generated using the parameter values that are unique across all test points. The user can use the buttons below the list box to reorder the test points, remove test points, and add test points. Test points can be added in one of three ways using the drop down menu.

Single

In this setting the test point is added 'as is' using the following syntax:

PARAMETER1:VALUE1,PARAMETER2:VALUE2,...

where VALUE can be either a number, string, or list of numbers/strings. Note, unlike in multiple, if a list is given the list will be written as is to the test point and not enumerated.

Multiple

In this setting multiple test points are added based on an input list using the following syntax:

PARAMETER1:[a1,b1,c1,d1,...],PARAMETER2:[a2,b2,c2,d2,...],...

Each value in the list is added as a single test point for its given parameter.

Combination

In this setting unique combinations of all inputs are added as separate test points. The input syntax is the same as in Multiple.

Parameters [4]

The parameters list box shows all the parameters and their values for the highlighted test point. These can be updated using the entry field below the list box. Select the parameter, enter a new value in the entry field, press update.

General Flow

There will generally be two paths that the user will follow when creating a test matrix: 1) starting from scratch, 2) starting with an existing matrix. When starting from scratch, the following steps will generally be used:

1. Give the test a name in the entry field at the top of the main GUI.
2. Open the parameters window.
    1. Load an existing parameters template or add the desired test parameters and their default values and flags.
3. (Optional) Open the the definitions window. If no definitions are required, skip to step 6.
    1. Load an existing definitions template or add the desired definitions and their meanings and flags.
4. Create a new test point group and select it.
5. Add test points to the group and modify as needed.
6. Repeat steps 4 through 5 as needed.
7. Press the save button to save the test matrix.
8. (Optional) Open the Plot Window to check the test point space.
9. When ready to output the final test matrix, selected the desired output format in the Formatted Output menu and press save.

When starting with an existing matrix, the user may choose to start at any of the above listed steps.

Appendix

TestMatrix Class

The main elements of TestMatrix class are detailed here so that the user can understand the layout when creating a new save script or any other custom scripts. The main elements are as follows:

1. GroupNames
   This is list of stored group name strings.
   For example: ['Group1','Group2',...]
2. GroupTestPoints
   This is a list of Pandas DataFrames. The order of this list corresponds to the order of the GroupNames list. This is where the test matrix is stored. The columns of the DataFrames are the test parameters, and the order is given by the Parameters dictionary.
3. Parameters
   This is a dictionary of dictionaries where the parameter names and their default values, flags , and timing data are stored as sub dictionaries to the parameter name. The value, flag, and timing keys are protected as 'VALUE' , 'FLAG', and 'TIMING'.
   For example: {'Param1:{'VALUE':'x','FLAG':'-','TIMING':{'VALUE':'y','FLAG':'D'}},...}
4. Definitions
   This is a dictionary of dictionaries that follows the same format as Parameters dictionary but excludes the timing subdictionary.