test_synthesis_nbook

September 29, 2018

```
In [ ]: import glob
        import json
        import pandas as pd
```

Let us gather the name of all json files into a list

```
In [2]: description_files = glob.glob('../*/test_*.json')

        print("The following test description files : ")
        print(description_files)
        print("will be imported")
```

```
The following test description files :
['../test_validation2DWaveSystemUpwindDeformedQuadrangles/test_WaveSystem2DUpwind_S
will be imported
```

Each json file content will be imported into a python dict, all these dict will be gathered into a list called `all_descriptions`

```
In [3]: # Let's import all json files into a list of dictionaries
        all_descriptions = []

        for file_name in description_files:
            with open(file_name, 'r') as fd:
                all_descriptions.append(json.load(fd))

        print("json files have been imported")
```

```
json files have been imported
```

In order to print a list (or a sublist), we need to import the pprint python package

```
In [4]: import pprint as pp
        #pp.pprint(all_descriptions)
```

```
In [5]: # Let's create a pandas dataframe out of our dict list
        df = pd.DataFrame(all_descriptions)
        print("The pandas dataframe has been created")

        list_of_all_columns = df.columns
        print("Printing the columns of the dataframe : these are the parameters of
        pp.pprint(list_of_all_columns)
```

```
The pandas dataframe has been created
Printing the columns of the dataframe : these are the parameters of the database
Index([u'Absolute_error', u'Boundary_conditions',
       u'Computational_time_taken_by_run', u'Geometry', u'Global_comment',
       u'Global_name', u'Initial_data', u'Linear_solver_algorithm',
       u'Linear_solver_maximum_iterations', u'Linear_solver_precision',
       u'Linear_solver_preconditioner', u'Linear_solver_with_scaling',
       u'Linear_system_max_actual_condition number',
       u'Linear_system_max_actual_error',
       u'Linear_system_max_actual_iterations_number', u'Mesh_cell_type',
       u'Mesh_dimension', u'Mesh_is_unstructured',
       u'Mesh_max_number_of_neighbours', u'Mesh_number_of_elements',
       u'Mesh_type', u'Numerical_method_name',
       u'Numerical_method_space_discretization',
       u'Numerical_method_time_discretization', u'Numerical_parameter_cfl',
       u'Numerical_parameter_space_step', u'Numerical_parameter_time_step',
       u'PDE_is_stationary', u'PDE_model',
       u'PDE_search_for_stationary_solution',
       u'Part_of_mesh_convergence_analysis', u'Relative_error',
       u'Simulation_final_number_of_time_steps_after_run',
       u'Simulation_final_time_after_run', u'Simulation_output_frequency',
       u'Simulation_parameter_maximum_time',
       u'Simulation_parameter_maximum_time_step', u'Space_dimension',
       u'Test_color'],
      dtype='object')
```

```
In [6]: #print("Printing the dataframe")
        #pp.pprint(df)

        # print values of columns: the name of the column can be used as attributes
        #print("Values of the Name column")
        #print(df.Global_name)
```

```
In [7]: # a new dataframe with a few columns only
        #Tous les résultats avec cfl >1
        column_list = ['Geometry', 'Numerical_parameter_cfl']
        sub_df1 = df[column_list]
        #print("sub_df1")
        #pp.pprint(sub_df1)
```

```
In [8]: # a new dataframe according to the CFL value
        sub_df2 = df[df.Numerical_parameter_cfl > 0.1]
        #print("sub_df2")
        #pp.pprint(sub_df2)
```

```
In [9]: # sorting a dataframe
        df.sort_values(by=['Global_name', 'Numerical_parameter_cfl'], ascending=Tru

        sub_df3 = df[df['Boundary_conditions'].isin(['Dirichlet'])]
        #print("sub_df3")
        #pp.pprint(sub_df3)
```

1 Displaying validation test tables with qgrid

Let's play with `qgrid now.` First extract the most interesting columns and visualise them in a widget.

```
In [10]: import qgrid

         # here's a cool dictionnary of options for displaying data
         gopt={
             'fullWidthRows': True,
             'syncColumnCellResize': True,
             'forceFitColumns': True,
             'defaultColumnWidth': 150,
             'rowHeight': 28,
             'enableColumnReorder': True,
             'enableTextSelectionOnCells': True,
             'editable': False,
             'autoEdit': False,
             'explicitInitialization': True,
             'maxVisibleRows': 40,
             'minVisibleRows': 8,
             'sortable': True,
             'filterable': True,
             'highlightSelectedCell': False,
             'highlightSelectedRow': True
         }

         # Extract the most interesting column from df into a second dataframe df2
         df2=df[['PDE_model','Numerical_method_name','Mesh_dimension','Mesh_type','

         # Let's create a jupyter table widget from the dataframe df2
         qgrid_widget=qgrid.show_grid(df2, grid_options=gopt, show_toolbar=False)

         # let's output this widget
         qgrid_widget
```

```
QgridWidget(grid_options={'defaultColumnWidth': 150, 'highlightSelectedRow': True,
```

2 Exporting validation test table to CSV and Excel format

`pandas` can be used to export to csv and excel, this is useful!

Let us first export the large database df.

```
In [11]: df.to_csv('test_synthesis_all.csv')#Saving using csv format
         output_file_name='test_synthesis_all.xlsx'
         writer = pd.ExcelWriter(output_file_name)
         df.to_excel(writer,'Sheet1')
         writer.save()
         print("Done writing file "+output_file_name)
```

```
Done writing file test_synthesis_all.xlsx
```

Let us now export the short database df2.

```
In [12]: df2.to_csv('test_synthesis_short.csv')#Saving using csv format
         output_file_name='test_synthesis_short.xlsx'
         writer = pd.ExcelWriter(output_file_name)
         df2.to_excel(writer,'Sheet1')
         writer.save()
         print("Done writing file "+output_file_name)
```

```
Done writing file test_synthesis_short.xlsx
```

```
In [13]: ls
```

```
CMakeFiles/          Makefile                 test_synthesis_nbook.ipynb
cmake_install.cmake  test_synthesis_all.csv   test_synthesis_short.csv
CTestTestfile.cmake  test_synthesis_all.xlsx  test_synthesis_short.xlsx
```

3 Convergence study table

```
In [14]: convergence_files = glob.glob('../*/Convergence_*.json')

         print("The following convergence description files : ")
         print(description_files)
         print("will be imported")
```

```
The following convergence description files :
['../test_validation2DWaveSystemUpwindDeformedQuadrangles/test_WaveSystem2DUpwind_S
will be imported
```

```
In [15]: # Let's import all json files into a list of dictionaries
         convergence_descriptions = []

         for file_name in convergence_files:
             with open(file_name, 'r') as fd:
                 convergence_descriptions.append(json.load(fd))

         print("convergence json files have been imported")
```

```
convergence json files have been imported
```

```
In [16]: # Let's create a pandas dataframe out of the convergence dictionary
         df_convergence = pd.DataFrame(convergence_descriptions)
         print("The convergence pandas dataframe has been created")

         list_of_all_columns_convergence = df_convergence.columns
         print("Printing the columns of the dataframe : these are the parameters of
         pp.pprint(list_of_all_columns_convergence)
```

```
The convergence pandas dataframe has been created
Printing the columns of the dataframe : these are the parameters of the database
Index([u'Boundary_conditions', u'Color', u'Computational_time',
       u'Condition_numbers', u'Errors', u'Final_time', u'Final_time_step',
       u'Geometry', u'Global_comment', u'Global_name', u'Initial_data',
       u'Max_vel_norm', u'Mesh_cell_type', u'Mesh_description',
       u'Mesh_dimension', u'Mesh_is_unstructured', u'Mesh_names', u'Mesh_path',
       u'Mesh_sizes', u'Mesh_type', u'Numerical_error_pressure',
       u'Numerical_error_velocity', u'Numerical_method_name',
       u'Numerical_method_space_discretization',
       u'Numerical_method_time_discretization', u'Numerical_parameter_cfl',
       u'PDE_is_stationary', u'PDE_model',
       u'PDE_search_for_stationary_solution',
       u'Part_of_mesh_convergence_analysis', u'Scaling_preconditioner',
       u'Scheme_order', u'Scheme_order_press', u'Scheme_order_vel',
       u'Space_dimension', u'Test_color'],
      dtype='object')
```

```
In [17]: # Extract the most interesting column from df_convergence into a second da
         df2_convergence=df_convergence[['PDE_model','Numerical_method_name','Mesh_

         # Let's create a jupyter table widget from the convergenc dataframe
         qgrid_widget_convergence=qgrid.show_grid(df2_convergence, grid_options=gop

         # let's output this widget
         qgrid_widget_convergence
```

```
QgridWidget(grid_options={'defaultColumnWidth': 150, 'highlightSelectedRow': True,
In [18]: #Now export convergence study table
         df_convergence.to_csv('Convergence_table_all.csv')#Saving using csv format
         output_file_name='Convergence_table_all.xlsx'
         writer = pd.ExcelWriter(output_file_name)
         df_convergence.to_excel(writer,'Sheet1')
         writer.save()
         print("Done writing file "+output_file_name)

         df2_convergence.to_csv('Convergence_table_short.csv')#Saving using csv for
         output_file_name='Convergence_table_short.xlsx'
         writer = pd.ExcelWriter(output_file_name)
         df2_convergence.to_excel(writer,'Sheet1')
         writer.save()
         print("Done writing file "+output_file_name)
```

```
Done writing file Convergence_table_all.xlsx
Done writing file Convergence_table_short.xlsx
```