convergence_synthesis_StationaryDiffusion

September 19, 2019

In [1]:
```
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 :
['../2DFV_Neumann_RightTriangles/test_Poisson2D_FV_20000Cells_Neumann_Regular_RightTriangles.js
```

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 the databas
        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'Mesh_cell_type', u'Mesh_dimension',
       u'Mesh_is_unstructured', u'Mesh_number_of_elements',
       u'Numerical_method_name', u'Numerical_method_space_discretization',
       u'Numerical_method_time_discretization', u'PDE_is_stationary',
       u'PDE_model', u'PDE_search_for_stationary_solution',
       u'Part_of_mesh_convergence_analysis', u'Relative_error',
       u'Space_dimension'],
      dtype='object')
```

```
In [73]: #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 [74]: # a new dataframe with a few columns only
         #Tous les résultats avec cfl >1
         column_list = ['Geometry', 'Boundary_conditions']
         sub_df1 = df[column_list]
         #print("sub_df1")
         #pp.pprint(sub_df1)
```

```
In [75]: # sorting a dataframe
         df.sort_values(by=['PDE_model','Numerical_method_name','Mesh_dimension','Mesh_cell_ty

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

```
---------------------------------------------------------------------------

TypeError                                 Traceback (most recent call last)

<ipython-input-75-fcc95fb632fa> in <module>()
```

```
      1 # sorting a dataframe
----> 2 df.sort_values(by=['PDE_model','Numerical_method_name','Mesh_dimension','Mesh_cell
      3
      4 sub_df3 = df[df['Boundary_conditions'].isin(['Dirichlet'])]
      5 #print("sub_df3")


    /usr/lib64/python2.7/site-packages/pandas/core/frame.pyc in sort_values(self, by, axis
   4412                 keys.append(k)
   4413             indexer = lexsort_indexer(keys, orders=ascending,
-> 4414                                       na_position=na_position)
   4415             indexer = _ensure_platform_int(indexer)
   4416         else:


    /usr/lib64/python2.7/site-packages/pandas/core/sorting.pyc in lexsort_indexer(keys, or
    205         # create the Categorical
    206         else:
--> 207             c = Categorical(key, ordered=True)
    208
    209         if na_position not in ['last', 'first']:


    /usr/lib64/python2.7/site-packages/pandas/core/arrays/categorical.pyc in __init__(self
    345                 codes, categories = factorize(values, sort=True)
    346             except TypeError:
--> 347                 codes, categories = factorize(values, sort=False)
    348                 if dtype.ordered:
    349                     # raise, as we don't have a sortable data structure and so


    /usr/lib64/python2.7/site-packages/pandas/util/_decorators.pyc in wrapper(*args, **kwa
    176                 else:
    177                     kwargs[new_arg_name] = new_arg_value
--> 178             return func(*args, **kwargs)
    179         return wrapper
    180     return _deprecate_kwarg


    /usr/lib64/python2.7/site-packages/pandas/core/algorithms.pyc in factorize(values, sor
    628                                             na_sentinel=na_sentinel,
    629                                             size_hint=size_hint,
--> 630                                             na_value=na_value)
    631
    632     if sort and len(uniques) > 0:


    /usr/lib64/python2.7/site-packages/pandas/core/algorithms.pyc in _factorize_array(valu
```

```
    474     uniques = vec_klass()
    475     labels = table.get_labels(values, uniques, 0, na_sentinel,
--> 476                               na_value=na_value)
    477
    478     labels = _ensure_platform_int(labels)


pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get


TypeError: unhashable type: 'list'
```

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 [6]:
```
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_cell_type','Mesh_nu

# 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
```

UWdyaWRXaWRnZXQoZ3JpZF9vcHRpb25zPXsnZGVmYXVsdENvbHVtbldpZHRoJzogMTUwLCAnaGlnaGxpZ2h0U2VsZWN0ZWI

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

```
Done writing file test_synthesis_StationaryDiffusion_all.xlsx
```

Let us now export the short database df2.

```
In [8]: df2.to_csv('test_synthesis_StationaryDiffusion_short.csv')#Saving using csv format
        df2.to_latex('test_synthesis_StationaryDiffusion_short.tex')#Saving using latex format
        output_file_name='test_synthesis_StationaryDiffusion_short.xlsx'
        writer = pd.ExcelWriter(output_file_name)
        df2.to_excel(writer,'Sheet1')
        writer.save()#Saving using excel format
        print("Done writing file "+output_file_name)
```

```
Done writing file test_synthesis_StationaryDiffusion_short.xlsx
```

3 Convergence study table

```
In [9]: convergence_files = glob.glob('../*/Convergence_*.json')

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

```
The following convergence description files :
['../2DFV_Neumann_RightTriangles/Convergence_Poisson_2DFV_squareWithTriangles.json', '../2DFV_
will be imported
```

```
In [10]: # 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 [11]:
```
        # 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")
        df_convergence.sort_values(by=['PDE_model','Numerical_method_name','Mesh_dimension','I

        list_of_all_columns_convergence = df_convergence.columns
        print("Printing the columns of the dataframe : these are the parameters of the databas
        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'Computational_time', u'Errors', u'Initial_data', u'Mesh_cell_type',
       u'Mesh_description', u'Mesh_dimension', u'Mesh_is_unstructured',
       u'Mesh_names', u'Mesh_sizes', u'Mesh_type', u'Numerical_method_name',
       u'Numerical_method_time_discretization', u'PDE_is_stationary',
       u'PDE_model', u'PDE_search_for_stationary_solution',
       u'Part_of_mesh_convergence_analysis', u'Scheme_order',
       u'Space_dimension', u'Test_color'],
      dtype='object')
```

In [12]:
```
        # Extract the most interesting column from df_convergence into a second dataframe df2
        df2_convergence=df_convergence[['PDE_model','Numerical_method_name','Mesh_dimension',

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

        # let's output this widget
        qgrid_widget_convergence
```

```
UWdyaWRXaWRnZXQoZ3JpZF9vcHRpb25zPXsnZGVmYXVsdENvbHVtbldpZHRoJzogMTUwLCAnaGlnaGxpZ2h0U2VsZWN0ZWR
```

In [13]:
```
        #Now export convergence study table
        df_convergence.to_csv('convergence_synthesis_StationaryDiffusion_all.csv')#Saving usi
        output_file_name='convergence_synthesis_StationaryDiffusion_all.xlsx'
        writer = pd.ExcelWriter(output_file_name)
        df_convergence.to_excel(writer,'Sheet1')
        writer.save()#Saving using excel format
```

```
        print("Done writing file "+output_file_name)

        df2_convergence.to_csv('convergence_synthesis_StationaryDiffusion_short.csv')#Saving
        df2_convergence.to_latex('convergence_synthesis_StationaryDiffusion_short.tex')#Savin
        output_file_name='convergence_synthesis_StationaryDiffusion_short.xlsx'
        writer = pd.ExcelWriter(output_file_name)
        df2_convergence.to_excel(writer,'Sheet1')
        writer.save()#Saving using excel format
        print("Done writing file "+output_file_name)
```

```
Done writing file convergence_synthesis_StationaryDiffusion_all.xlsx
Done writing file convergence_synthesis_StationaryDiffusion_short.xlsx
```