Grid*Pro* API – Quality Control

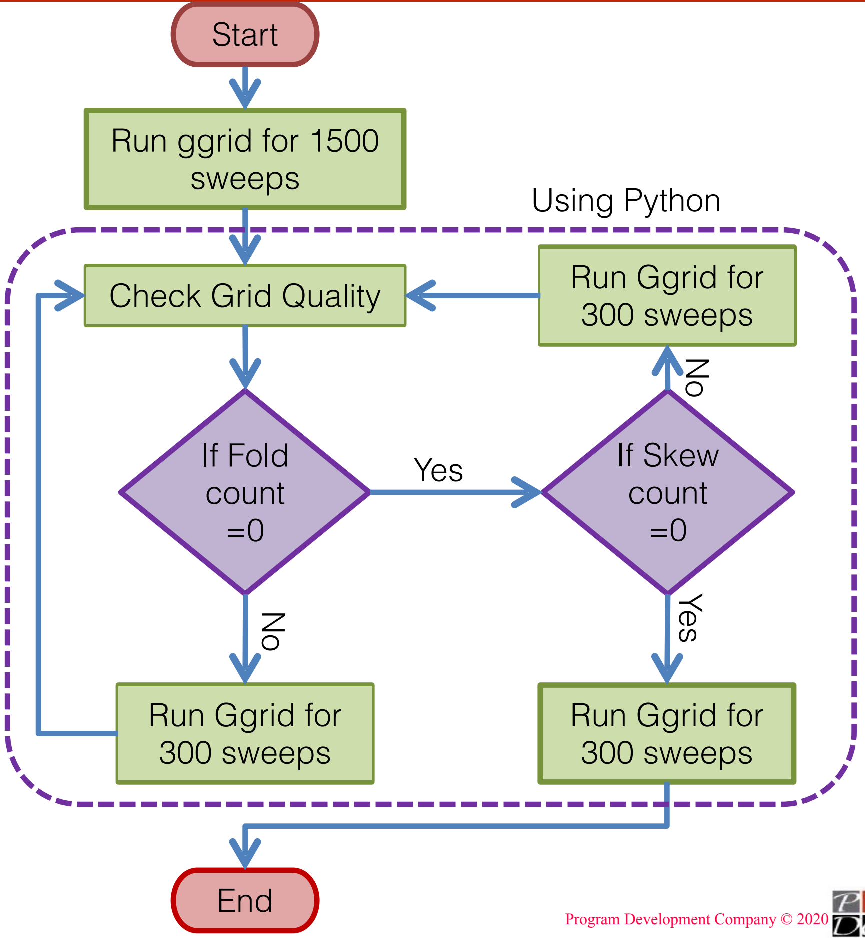
API Tutorial - 3

Scope of the tutorial

- In API - Schedule File tutorial, we discussed about how to create a schedule file to control the grid generator.
- In this tutorial, we will discuss about how to control the grid generator using the quality parameters.
- Schedule file has many features, one such is to run system command while running the grid generator.
- We are going to exploit that option along with the other command line tools in order to achieve the goal.
- The advanced functions of the schedule file are not exposed, hence we will be using the command line utilities instead of API.

Problem Definition

- The work flow of the script is as follows:
  1. Run Ggrid for 5 counts with 300 sweeps for every count
  2. Check grid quality. i.e if surface and volume fold are 0 and the desired skew is achieved.
  3. If not, run ggrid for another 300 sweeps.
  4. Repeat 2 and 3 till the desired values are achieved.
  5. Once achieved run Ggrid for another count of 300 sweeps to additionally smooth the grid in the regions where the blocks were skewed.
  6. End



Input Parameters

- To start with, we need to provide a schedule file using the input parameters.
- We will run ggrid for 5 steps with 300 sweeps for every step.
- Next we will check the quality of the grid using system command.
- The desired quality parameters are folds and skew. We can also add aspect ratio and warpage in the future.
- Run python script to compare the quality parameters.

Input

- ☐ Topology - step5.final_topo
- ☐ Step Count - 5
- ☐ Sweep Count - 300
- ☐ Output - volute.grd
- ☐ Folds – 0
- ☐ Skewness – 0.8

```
step 5: -c all 1.0 0 -C all 1.0 24 -r -S 300 -w
step 6: -sys 'ws qchk volute.grd 11 10000 0.8 120'
step 7: -sys 'python Quality.py step5.final_topo.sch'
write -f volute.grd
```

Figure 1: Input Schedule file

GridPro

Quality Check

Command name: `qchk [options]`

Syntax: *`qchk <grid file name> <Minimum no. of bad volumes(0 to 11)> <Aspect ratio threshold (1 to Inf)> <Skewness threshold(0 to 1)> <Warpage Threshold(0 to 180)>`*

Example:
```
ws qchk volute.grd 11 10000 0.8 120
```

Note:

1. To check no. of folds/negative volumes, we need to input 11 to get the same result as in GUI.
2. The output would be written to 4 hex files, One for each parameter→ bad_folds.hex, bad_skewness.hex, bad_asp_ratio.hex, bad_warp.hex

Code Snippet

```python
# Import libraries
import sys
import fileinput

# Evaluate fold count and skewness value from qcheck output files
def evaluate_fold_count_from_qcheck_output():
    # Calculating number of folds and skew
    folds = open('bad_folds.hex').readline()
    folds = int(folds)
    skew = open('bad_skewness.hex').readline()
    skew = int(skew)
    return [folds, skew]

# Extend schedule file if the grid quality is not good enough
def extend_schedule_file(schedule_file_name):

    # Input Parameters
    step_count = 5
    sweep_count = 300
    skewness = 0.8

    # Evaluate fold count and skewness value from qcheck output files
    folds, skew = evaluate_fold_count_from_qcheck_output()

    # Checking the desired quality condition
    count = step_count-1
    is_good_enough = (folds == 0 and skew == 0)
    for line in fileinput.input(schedule_file_name, inplace=1):
        count += 1
        if count > 50:
            break
        elif line.startswith('write'):
            if is_good_enough:
                print ("step {}: -c all 1.0 0 -C all 1.0 24 -r -S {} -w\n" + line.rstrip()).format(count, sweep_count)
            else:
                print ("step {}: -c all 1.0 0 -C all 1.0 24 -r -S {} -w "
                       "\nstep {}: -sys 'ws qchk volute.grd 11 10000 {} 120' "
                       "\nstep {}: -sys 'python Quality.py {}'\n"
                       + line.rstrip()).format(count, sweep_count, count + 1, skewness, count + 2, schedule_file_name)
        else:
            print line.rstrip()


# Main Function
if (__name__ == '__main__'):
    extend_schedule_file(sys.argv[1])
```

API Module

- Import the following libraries:
  1. sys = To read command line input
  2. fileinput = To iterate over lines from multiple input streams

```python
import sys
import fileinput
```

- First Function: To evaluate the fold and skewness from the quality check output files
  1. Read the first line of bad_folds.hex and bad_skewness.hex
  2. Return the output as integers

```python
def evaluate_fold_count_from_qcheck_output():
    folds = open('bad_folds.hex').readline()
    folds = int(folds)
    skew = open('bad_skewness.hex').readline()
    skew = int(skew)
    return [folds, skew]
```

Note: These two values would be zero if it satisfies the desired grid quality else the number of folded and skewed cells would be listed.

API Module

- Creating a second function(extend_schedule_file) where we take the input parameters such as desired grid quality, sweep count & step count and check if the desired grid quality is achieved.

- Enter the input parameters

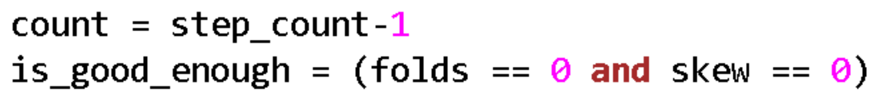
```
def extend_schedule_file(schedule_file_name):

    # Input Parameters
    step_count = 5
    sweep_count = 300
    skewness = 0.8
```

- Run the first function to get the number of folds and skewed cells above the desired quality.

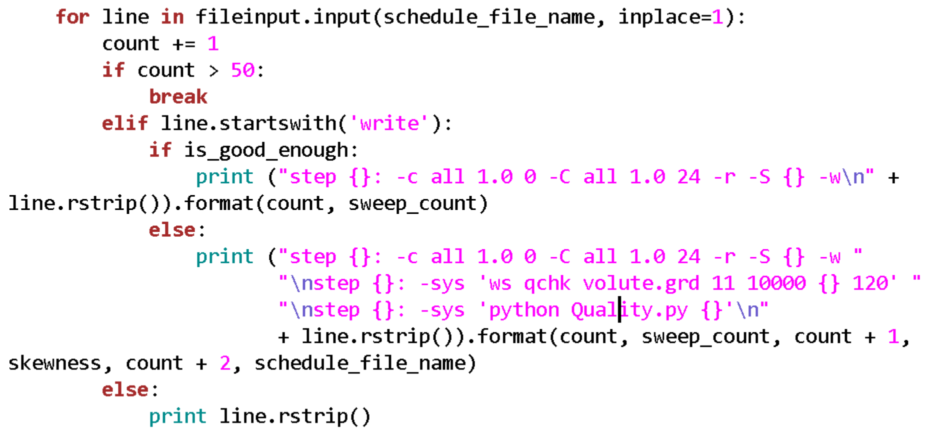
```
folds, skew = evaluate_fold_count_from_qcheck_output()
```

API Module

- Count refers to the no. of steps mentioned in the schedule file.
- Checking if the grid quality is as desired.
- If the desired grid quality is achieved, then add a line to the schedule file to run final number of sweeps and stop.
- If not, then add lines to continue the number of sweeps and redo the quality check. Continue this loop until the desired quality is achieved.
- The loop automatically breaks if the no. of steps reach 50.



```
count = step_count-1
is_good_enough = (folds == 0 and skew == 0)
```



```
    for line in fileinput.input(schedule_file_name, inplace=1):
        count += 1
        if count > 50:
            break
        elif line.startswith('write'):
            if is_good_enough:
                print ("step {}: -c all 1.0 0 -C all 1.0 24 -r -S {} -w\n" +
line.rstrip()).format(count, sweep_count)
            else:
                print ("step {}: -c all 1.0 0 -C all 1.0 24 -r -S {} -w "
                       "\nstep {}: -sys 'ws qchk volute.grd 11 10000 {} 120' "
                       "\nstep {}: -sys 'python Quality.py {}'\n"
                       + line.rstrip()).format(count, sweep_count, count + 1,
skewness, count + 2, schedule_file_name)
        else:
            print line.rstrip()
```

API Module

- Finally call the extend_schedule_file function under the main function to run the script.
- This function in turn calls the evaluate_fold_count_from_qcheck_output() function to compare the quality.

```
if (__name__ == '__main__'):
    extend_schedule_file(sys.argv[1])
```

End of the Tutorial