User Guide for autotune_nightly.py

Carolyn Kao (chkao831@stanford.edu)
Mentors: Jerry Watkins & Irina Tezaur (Sandia Nat'l Labs)

May 28, 2021

1 Program Description

- This script currently serves as part of the scripts that are running on a nightly basis for post-processing[1]. Its primary purposes are to run an optimization algorithm, update an input file and record history so that the next nightly test will run a new iteration based on the updated input file. The corresponding outputs (timers, etc.) would be automatically extracted and recorded to the history file the next time calling this script, along with new input information for the next nightly test.
- This version utilizes random search in regards to the choice of parameters, i.e. each iteration is completely random from the previous ones.
- The current infrastructure focuses on tuning parameters from the `ParameterList` for two `mySmoother`s.
- Codes and relevant files are available at `https://github.com/chkao831/Autotuning/tree/main/autotune_nightly`

2 Parameter Space

We currently focus on tuning parameters for two `mySmoother`s at a time. The input files are multiple-level nested YAML files, in which `mySmoother`s are represented in the form of

```yaml
mySmoother1:
    factory:
    type: [TUNING FOCUS]
    'smoother: pre or post':
    ParameterList: [TUNING FOCUS]
mySmoother4:
    factory:
    type: [TUNING FOCUS]
    'smoother: pre or post':
    ParameterList: [TUNING FOCUS]
```

[1]The real-time nightly tests produce a ctest json file that locates at either here: `https://github.com/ikalash/ikalash.github.io/tree/master/ali/blake_nightly_data` (CPU) or `https://github.com/ikalash/ikalash.github.io/tree/master/ali/weaver_nightly_data` (GPU)

The options that we enable for each smoothers are the following,

```

type: RELAXATION
ParameterList:
    'relaxation: type': MT Gauss-Seidel
    'relaxation: sweeps': positive integer
    'relaxation: damping factor': positive real number

type: RELAXATION
ParameterList:
    'relaxation: type': Two-stage Gauss-Seidel
    'relaxation: sweeps': positive integer
    'relaxation: inner damping factor': positive real number

type: CHEBYSHEV
ParameterList:
    'chebyshev: degree': positive integer
    'chebyshev: ratio eigenvalue': positive real number
    'chebyshev: eigenvalue max iterations': positive integer
```

3 User Interface

The user would need to specify the input filename, the casename, the optimization algorithm, and the possible choices with ranges for each smoother in a file. A snapshot of a sample file called _properties.json follows,

```
 "input": "input_albany_Velocity_MueLuKokkos_Wedge.yaml",
 "case": "humboldt-3-20km_vel_muk_wdg_tune_np1",
 "algorithm": "random_search",
 "mS1": {
     "type_options": ["Two-stage Gauss-Seidel", "MT Gauss-Seidel"], # CHEBYSHEV is disabled for mS1
     "relaxation: sweeps": {
         "inclusive_lower_bound": 1,
         "inclusive_upper_bound": 2
     },
     "relaxation: damping factor": {
     # for MT Gauss-Seidel
             "mean": 1,
             "sd": 0.1,
             "low": 0.8,
             "upper": 1.2
     },
     "relaxation: inner damping factor": {
     # for Two-stage Gauss-Seidel
             "mean": 1,
             "sd": 0.1,
             "low": 0.8,
             "upper": 1.2
     },
     "chebyshev: degree": {
     # since CHEBYSHEV is disabled for mS1, this is not used
             "inclusive_lower_bound": 1,
```

```
27                "inclusive_upper_bound": 6
28        },
29        "chebyshev: ratio eigenvalue": {
30        # not used
31                "mean": 30,
32                "sd": 15,
33                "low": 10,
34                "upper": 50
35        },
36        "chebyshev: eigenvalue max iterations": {
37        # not used
38                "inclusive_lower_bound": 5,
39                "inclusive_upper_bound": 100
40        }
41    }
42    # "mS4" omitted due to the space constraint
```

Note that on `line 5`, only two types are listed – `type: CHEBYSHEV` is disabled for `mySmoother1`. Hence, the information from `line 24-38` is not used (although they could still be listed). That being said, the user could also open up all three choices or simply enable only one choice.

If the specified `type_options` in the `_properties.json` file contains some string other than `["Two-stage Gauss-Seidel", "MT Gauss-Seidel", "CHEBYSHEV"]`, a `ValueError("Type options are not defined by MT Gauss-Seidel, Two-stage Gauss-Seidel or CHEBYSHEV")` would be raised.

In terms of the range, `inclusive_lower_bound` and `inclusive_upper_bound` define the range for integers within `[inclusive_lower_bound, inclusive_upper_bound]`, inclusively.

In addition, `mean`, `sd`, `low`, and `upper` define the range for real numbers derived from that of a normally distributed continuous random variable, truncated to the range [`low`, `upper`], with distribution mean and standard deviation of `mean` and `sd` respectively.

4 Prerequisites

In addition to the properties file such as `_properties.json`, under the working directory, one needs

- An input yaml file that is listed in `_properties.json`
- ctest output files, starting from the second run
- to pip install `pandas, ruamel.yaml, and scikit-learn`

5 Command Line Usage

```
$ python3 autotune_nightly.py --help
```

```
usage: autotune_nightly.py [-h] properties_file
    ctest_output_file

positional arguments:
  properties_file    parameter space definition (.json)
  ctest_output_file  ctest output filename (.json)

optional arguments:
  -h, --help         show this help message and exit
```

For example, with
`$ python3 autotune_nightly.py _properties.json ctest-20210520.json`,
the 2nd argument `_properties.json` points to the properties file; the 3rd argument `ctest-20210520.json` specifies the output file from which the output information (such as timers) are extracted for evaluation.

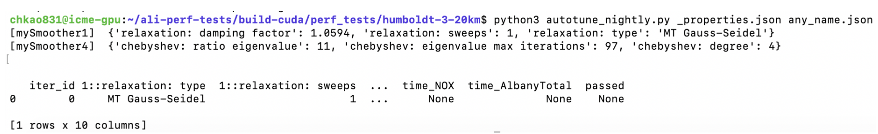
However, for the first iteration of this script (`#iter_id`=0), the ctest argument is useless, as there's no nightly test run yet. In this case, the user could input any filename with .json extension to this required argument, but whatever it is, the argument would not be used.

If the specified `case` in the `_properties.json` file cannot be found from the ctest output file that is called in command line, a `ValueError: The casename is not found in the ctest output file` would be raised.

6 Simulation

For the first run (`#iter_id`=0), as there's no history file available, the script would simply update the input yaml file with a set of updated parameters by the optimization algorithm. By the end of this iteration, a history file `[case]_hist.csv` is created and the updated input parameters are written to file. Meanwhile, a copy of the input yaml file is saved, called `[input]_0.yaml`, where `input` corresponds to the yaml filename from `_properties.json` and 0 specifies the `#iter_id`. An example is as follows,

```
$ python3 autotune_nightly.py _properties.json any_name.json
```


```
chkao831@icme-gpu:~/ali-perf-tests/build-cuda/perf_tests/humboldt-3-20km$ python3 autotune_nightly.py _properties.json any_name.json
[mySmoother1]  {'relaxation: damping factor': 1.0594, 'relaxation: sweeps': 1, 'relaxation: type': 'MT Gauss-Seidel'}
[mySmoother4]  {'chebyshev: ratio eigenvalue': 11, 'chebyshev: eigenvalue max iterations': 97, 'chebyshev: degree': 4}


   iter_id 1::relaxation: type  1::relaxation: sweeps  ...  time_NOX  time_AlbanyTotal  passed
0        0     MT Gauss-Seidel                      1  ...      None              None    None

[1 rows x 10 columns]
```

A dataframe is printed to the command line, as illustrated above. The last three entries, `time_NOX, time_AlbanyTotal` and `passed`, are temporarily set to `None` by the end of this iteration, since no nightly test is run yet.

`time_NOX` is defined as `NOX Total Linear Solve + NOX Total Preconditioner Construction`. `time_AlbanyTotal` is the `Albany Total Time` from the ctest output file.

Then, for the next-day iteration, given that the history file `[case]_hist.csv` preexists, an output ctest json file should have been generated based on the previously-updated input file. As an example, let the output ctest file be `ctest-20210520.json`,

```
$ python3 autotune_nightly.py _properties.json ctest
   -20210520.json
```

```
chkao831@icme-gpu:~/ali-perf-tests/build-cuda/perf_tests/humboldt-3-20km$ python3 autotune_nightly.py _properties.json ctest-20210520.json

[mySmoother1]  {'relaxation: damping factor': 0.9559, 'relaxation: sweeps': 2, 'relaxation: type': 'MT Gauss-Seidel'}
[mySmoother4]  {'relaxation: inner damping factor': 0.9995, 'relaxation: sweeps': 2, 'relaxation: type': 'Two-stage Gauss-Seidel'}

   iter_id 1::relaxation: type  1::relaxation: sweeps  ...  time_NOX time_AlbanyTotal  passed
0        0     MT Gauss-Seidel                      1  ...   13.4905          34.1335    True
1        1     MT Gauss-Seidel                      2  ...      None             None    None

[2 rows x 13 columns]
```

From the first row of the dataframe, we could see that the timer data and a boolean that indicates test pass/fail of iteration 0 is extracted from `ctest-20210520.json`. At the same time, the updated input parameters are generated for the next round of nightly test.

This is repeated on a nightly basis.

7 Deliverables

7.1 [case]_hist.csv

Starting from the first run (`#iter_id`=0), a history file in csv format would be generated. The iteration id, updated input parameters, and corresponding output results are written to the file.

If the ctest does not pass for an experiment, the value of `inf` would be put to `time_NOX` and `time_AlbanyTotal` because not timer data is available for evaluation. In such cases, `passed = FALSE`.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	iter_id	1::relaxation	1::relaxation	1::relaxation	1::relaxation	4::relaxation	4::relaxation	4::relaxation	4::chebyshev	4::chebyshev	4::chebyshev	time_NOX	time_Albany	passed
2	0	MT Gauss-Se	1	1.0594					4	11	97	13.49047	34.1335	TRUE
3	1	MT Gauss-Se	2	0.9559		Two-stage G	2	0.9995				13.60902	34.4281	TRUE
4	2	MT Gauss-Se	2	0.9475					1	39	8	14.03035	33.1238	TRUE
5	3	Two-stage G	1		0.9386				5	49	29	12.28761	31.3359	TRUE
6	4	MT Gauss-Se	1	0.9031		Two-stage G	2	0.9363						

Figure 1: (Example) humboldt-3-20km_vel_muk_wdg_tune_np1_hist.csv

7.2 [case]_hist_sorted.csv

Starting from the second run (`#iter_id`=1), a history file in csv format would be generated. It contains the same information as `[case]_hist.csv` does, except that it differs in row order – instead of ordering by `#iter_id`=0, 1, 2..., the experiments are sorted in ascending order by `time_NOX`.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	iter_id	1::relaxation	1::relaxation	1::relaxation	1::relaxation	4::relaxation	4::relaxation	4::relaxation	4::chebyshev	4::chebyshev	4::chebyshev	time_NOX	time_Albany	passed
2	3	Two-stage G	1		0.9386				5	49	29	12.28761	31.3359	TRUE
3	0	MT Gauss-Se	1	1.0594					4	11	97	13.49047	34.1335	TRUE
4	1	MT Gauss-Se	2	0.9559		Two-stage G	2	0.9995				13.60902	34.4281	TRUE
5	2	MT Gauss-Se	2	0.9475					1	39	8	14.03035	33.1238	TRUE
6	4	MT Gauss-Se	1	0.9031		Two-stage G	2	0.9363						

Figure 2: (Example) humboldt-3-20km_vel_muk_wdg_tune_np1_hist_sorted.csv

From here, we see that the experiment with `#iter_id=3` achieves the best performance in terms of `time_NOX(=12.28761)`.

7.3 [input]_Best.yaml

At every iteration, we have checked to see if the current iteration is better than all past iterations. If true, we update `[input]_Best.yaml` with the inputs from the current iteration.

To verify that the output matches,

```
diff  input_albany_Velocity_MueLuKokkos_Wedge_3.yaml
    input_albany_Velocity_MueLuKokkos_Wedge_Best.yaml
```

This displays no difference in both files.

A Complete Command Line Sample

```
chkao831@icme-gpu:~/ali-perf-tests/build-cuda/perf_tests/humboldt-3-20km$ python3 autotune_nightly.py _properties.json any_name.json
[mySmoother1]  {'relaxation: damping factor': 1.0594, 'relaxation: sweeps': 1, 'relaxation: type': 'MT Gauss-Seidel'}
[mySmoother4]  {'chebyshev: ratio eigenvalue': 11, 'chebyshev: eigenvalue max iterations': 97, 'chebyshev: degree': 4}


   iter_id 1::relaxation: type  1::relaxation: sweeps  ...  time_NOX  time_AlbanyTotal  passed
0        0     MT Gauss-Seidel                      1  ...      None              None    None

[1 rows x 10 columns]
chkao831@icme-gpu:~/ali-perf-tests/build-cuda/perf_tests/humboldt-3-20km$ python3 autotune_nightly.py _properties.json ctest-20210520.json
[mySmoother1]  {'relaxation: damping factor': 0.9559, 'relaxation: sweeps': 2, 'relaxation: type': 'MT Gauss-Seidel'}
[mySmoother4]  {'relaxation: inner damping factor': 0.9995, 'relaxation: sweeps': 2, 'relaxation: type': 'Two-stage Gauss-Seidel'}


   iter_id 1::relaxation: type  1::relaxation: sweeps  ...  time_NOX time_AlbanyTotal  passed
0        0     MT Gauss-Seidel                      1  ...   13.4905          34.1335    True
1        1     MT Gauss-Seidel                      2  ...      None             None    None

[2 rows x 13 columns]
chkao831@icme-gpu:~/ali-perf-tests/build-cuda/perf_tests/humboldt-3-20km$ python3 autotune_nightly.py _properties.json ctest-20210521.json
[mySmoother1]  {'relaxation: damping factor': 0.9475, 'relaxation: sweeps': 2, 'relaxation: type': 'MT Gauss-Seidel'}
[mySmoother4]  {'chebyshev: ratio eigenvalue': 39, 'chebyshev: eigenvalue max iterations': 8, 'chebyshev: degree': 1}


   iter_id 1::relaxation: type  1::relaxation: sweeps  ...  time_NOX time_AlbanyTotal  passed
0        0     MT Gauss-Seidel                      1  ...   13.4905          34.1335    True
1        1     MT Gauss-Seidel                      2  ...    13.609          34.4281    True
2        2     MT Gauss-Seidel                      2  ...      None             None    None

[3 rows x 13 columns]
chkao831@icme-gpu:~/ali-perf-tests/build-cuda/perf_tests/humboldt-3-20km$ python3 autotune_nightly.py _properties.json ctest-20210523.json
[mySmoother1]  {'relaxation: inner damping factor': 0.9386, 'relaxation: sweeps': 1, 'relaxation: type': 'Two-stage Gauss-Seidel'}
[mySmoother4]  {'chebyshev: ratio eigenvalue': 49, 'chebyshev: eigenvalue max iterations': 29, 'chebyshev: degree': 5}


   iter_id     1::relaxation: type  1::relaxation: sweeps  ...  time_NOX  time_AlbanyTotal passed
0        0         MT Gauss-Seidel                      1  ...   13.4905           34.1335   True
1        1         MT Gauss-Seidel                      2  ...    13.609           34.4281   True
2        2         MT Gauss-Seidel                      2  ...   14.0304           33.1238   True
3        3  Two-stage Gauss-Seidel                      1  ...      None              None   None

[4 rows x 14 columns]
chkao831@icme-gpu:~/ali-perf-tests/build-cuda/perf_tests/humboldt-3-20km$ python3 autotune_nightly.py _properties.json ctest-20210524.json
[mySmoother1]  {'relaxation: damping factor': 0.9031, 'relaxation: sweeps': 1, 'relaxation: type': 'MT Gauss-Seidel'}
[mySmoother4]  {'relaxation: inner damping factor': 0.9363, 'relaxation: sweeps': 2, 'relaxation: type': 'Two-stage Gauss-Seidel'}


   iter_id     1::relaxation: type  1::relaxation: sweeps  ...  time_NOX  time_AlbanyTotal passed
0        0         MT Gauss-Seidel                      1  ...   13.4905           34.1335   True
1        1         MT Gauss-Seidel                      2  ...    13.609           34.4281   True
2        2         MT Gauss-Seidel                      2  ...   14.0304           33.1238   True
3        3  Two-stage Gauss-Seidel                      1  ...   12.2876           31.3359   True
4        4         MT Gauss-Seidel                      1  ...      None              None   None

[5 rows x 14 columns]
chkao831@icme-gpu:~/ali-perf-tests/build-cuda/perf_tests/humboldt-3-20km$ ls *.csv
humboldt-3-20km_vel_muk_wdg_tune_np1_hist.csv  humboldt-3-20km_vel_muk_wdg_tune_np1_hist_sorted.csv
chkao831@icme-gpu:~/ali-perf-tests/build-cuda/perf_tests/humboldt-3-20km$ diff  input_albany_Velocity_MueLuKokkos_Wedge_3.yaml input_albany
_Velocity_MueLuKokkos_Wedge_Best.yaml
chkao831@icme-gpu:~/ali-perf-tests/build-cuda/perf_tests/humboldt-3-20km$
```