Validating_python_version_from_cpp

May 18, 2020

1 Testing the python version against the c++ one

This notebook just run the python model with the same parameters than the c++ one and compare the results. This is to ensurem the python implementation is no different. To do so I am just using the Factor of safty: this metrics depends on all others and therefore validates the model.

The c++ test has been done with the `Test_param.param` parameter file and the `preprocessed_data.csv` input precipitation.

1.1 First ingesting the testing data

[1]:

```python
# Importing the model
import lsdfailtools.iverson2000 as iverson
# I'll need that to process the outputs
import pandas as pd
import numpy as np

# rainfall data
df = pd.read_csv("preprocessed_data.csv")

# Result time series from the c++ simulation
df_calib = pd.read_csv("mytestoutput_time_series_depth_FS.csv")
## Making sure I am using the same depths resolution
depths = df_calib.depth.values
```

1.2 Now running the model

Note that I just adjust the parameters that have a different defautl values than the c++ run

[2]:

```python
mymodel = iverson.iverson_model(alpha = 0.51, depths = depths)
mymodel.run(df.duration_s.values, df.intensity_mm_sec.values)
```

1.3 Comparing the datasets

`mymodel.cppmodel.output_FS_timedepth` and `df_calib` both show the same data in slightly different format: the time seris of the Factor of safety at depth. I am first checking if the values look similar, then the difference between the two

[3]:
```
print(mymodel.cppmodel.output_FS_timedepth[:,0])
print(df_calib["0.000000"].values)
```

```
[40.954075  20.698269  10.570364   7.194396   5.506412   4.493622
  3.8184283  3.3361468  2.974436   2.6931055  2.468041   2.2838974
  2.1304443  2.0005994  1.8893036  1.7928473  1.7084482  1.6339782
  1.5677829  1.5085554  1.4552506  1.4070225  1.3631787  1.3231475
  1.2864522  1.2526925  1.2215298  1.1926754  1.1658819  1.1409363
  1.1176538  1.0958734  1.0754542  1.0562725  1.0382192]
[40.9541   20.6983   10.5704    7.1944    5.50641   4.49362   3.81843   3.33615
  2.97444   2.69311   2.46804   2.2839    2.13044   2.0006    1.8893    1.79285
  1.70845   1.63398   1.56778   1.50856   1.45525   1.40702   1.36318   1.32315
  1.28645   1.25269   1.22153   1.19268   1.16588   1.14094   1.11765   1.09587
  1.07545   1.05627   1.03822]
```

[4]:
```
print(mymodel.cppmodel.output_FS_timedepth[:,60] - df_calib[df_calib.columns.
 ↪values[61] ].values)
```

```
[-1.59912109e-06 -1.55395508e-05 -2.00500488e-05 -3.18771362e-06
  1.20697021e-06 -4.59289551e-06 -4.24438477e-06 -2.66151428e-06
  1.71485901e-06  1.65710449e-06  4.40643311e-06 -3.37127686e-06
  2.21611023e-06  3.15021515e-06  4.20719147e-06  1.34616852e-06
  4.96715546e-06  1.33251190e-06  4.94510651e-06 -2.09148407e-06
 -4.59236145e-06  4.55432892e-06  3.99169922e-07  3.74496460e-06
 -2.57591248e-06 -3.37104797e-06 -3.84338379e-06 -1.03530884e-06
 -3.00777435e-06 -4.91996765e-06 -4.27452087e-06  3.47965240e-06
  2.68238068e-06  3.88595581e-06  1.70936584e-06]
```

Difference is unsignificant and due to floating points precision when converting into python

[ ]:
```

```