VdM analysis tutorial

July 2015

Monika Grothe (Wisconsin)

Where to find your data

- Whenever the LHC does an optimization scan with their automized tools, they send the information relevant for that scan in dedicated scan dip variables
- Among those variables, there is a scan acquisition flag
- For any period of time when this flag is true, Zhen's code dumps the content of the scan dip variables into a csv file vdm_<series of hex numbers>.csv
- These files can be found here:

```
[grothe@srv-C2C04-07 vdm]$ pwd
/brildata/vdm
[grothe@srv-C2C04-07 vdm]$ ls -l
total 6560
-rw-rw-r-- 1 xiezhen brilpro 2223322 Jul  9 11:41 dipdata_093db38f-7347-4f1a-b551-9f2265852db8.csv.tar.gz
-rw-rw-r-- 1 xiezhen brilpro  124199 Jul  9 04:00 vdm_093db38f-7347-4f1a-b551-9f2265852db8.csv
-rw-rw-r-- 1 xiezhen brilpro   77944 May 30 20:47 vdm_0da78634-da8d-4607-a0fa-03ecf84ca227.csv
-rw-rw-r-- 1 xiezhen brilpro   57513 Jul  8 00:07 vdm_1756adfb-7fb0-4104-8466-c930becfe878.csv
-rw-r--r-- 1 xiezhen brilpro 3343481 Jun  6 17:50 vdm_3829.csv
-rw-rw-r-- 1 xiezhen brilpro   67632 Jun 14 17:15 vdm_482f8ddd-7e67-4504-be47-9e17bc044394.csv
-rw-rw-r-- 1 xiezhen brilpro   61542 Jul 13 09:52 vdm_5e351062-03b9-46d1-b044-1ef15aaaf17b.csv
-rw-rw-r-- 1 xiezhen brilpro   92952 Jul  5 03:01 vdm_62ac871d-da34-4310-aba3-f0a215b8a97a.csv
-rw-rw-r-- 1 xiezhen brilpro   62319 Jul 10 19:58 vdm_6608b378-9d02-4d68-b769-daf45d065e8f.csv
-rw-rw-r-- 1 xiezhen brilpro  143299 Jun  7 22:36 vdm_6ec350f6-159a-4910-916d-907b72eec4fb.csv
-rw-rw-r-- 1 xiezhen brilpro   69497 May 28 18:29 vdm_80dcec94-7791-4499-b159-4e6c6d04eef1.csv
-rw-rw-r-- 1 xiezhen brilpro   89641 Jun  3 17:42 vdm_a68ed419-3c04-4522-911b-92504601c7ec.csv
-rw-rw-r-- 1 xiezhen brilpro   30835 Jul  7 04:09 vdm_ba71673c-8b3b-4e85-bd46-bc46064360ed.csv
-rw-rw-r-- 1 xiezhen brilpro   27437 May 26 17:11 vdm_d96f2df1-51e1-41c7-9370-7429995d08a3.csv
-rw-rw-r-- 1 xiezhen brilpro  154915 Jul 12 11:00 vdm_f63c64c2-b61a-4ddc-869e-803b6f1b5950.csv
```

Where to find your data (II)

- In the following, will step you through the scan done in fill 3976, in the early morning of July 9
- Have a look at vdm_093db38f-7347-4f1a-b551-9f2265852db8.csv

```
:fill,run,ls,nb,sec,msec,acqflag,step,beam,ip,scanstatus,plane,progress,nominal_separation,read_nominal_B1sepPlane,read_nominal_B1xingPlane,r
ead_nominal_B2sepPlane,read_nominal_B2xingPlane,set_nominal_B1sepPlane,set_nominal_B1xingPlane,set_nominal_B2sepPlane,set_nominal_B2xingPlane
3976,251252,480,60,1436406101,901,1,0,3,2,ACQUIRING,CROSSING,19,0,-0.01144,-0.16641,0.01152,0.16652,0,0,0,-0
3976,251252,480,64,1436406103,355,1,0,3,2,ACQUIRING,CROSSING,17,0,-0.01144,-0.16641,0.01152,0.16652,0,0,0,-0
3976,251252,481,4,1436406104,813,1,0,3,2,ACQUIRING,CROSSING,16,0,-0.01144,-0.16641,0.01152,0.16652,0,0,0,-0
3976,251252,481,8,1436406106,236,1,0,3,2,ACQUIRING,CROSSING,14,0,-0.01144,-0.16641,0.01152,0.16652,0,0,0,-0
3976,251252,481,12,1436406107,653,1,0,3,2,ACQUIRING,CROSSING,13,0,-0.01144,-0.16641,0.01152,0.16652,0,0,0,-0
3976,251252,481,16,1436406109,172,1,0,3,2,ACQUIRING,CROSSING,11,0,-0.01144,-0.16641,0.01152,0.16652,0,0,0,-0
3976,251252,481,20,1436406110,590,1,0,3,2,ACQUIRING,CROSSING,10,0,-0.01144,-0.16641,0.01152,0.16652,0,0,0,-0
3976,251252,481,24,1436406112,108,1,0,3,2,ACQUIRING,CROSSING,9,0,-0.01144,-0.16641,0.01152,0.16652,0,0,0,-0
3976,251252,481,28,1436406113,529,1,0,3,2,ACQUIRING,CROSSING,7,0,-0.01144,-0.16641,0.01152,0.16652,0,0,0,-0
3976,251252,481,32,1436406114,942,1,0,3,2,ACQUIRING,CROSSING,6,0,-0.01144,-0.16641,0.01152,0.16652,0,0,0,-0
3976,251252,481,36,1436406116,461,1,0,3,2,ACQUIRING,CROSSING,4,0,-0.01144,-0.16641,0.01152,0.16652,0,0,0,-0
3976,251252,481,40,1436406117,903,1,0,3,2,ACQUIRING,CROSSING,3,0,-0.01144,-0.16641,0.01152,0.16652,0,0,0,-0
3976,251252,481,44,1436406119,330,1,0,3,2,ACQUIRING,CROSSING,1,0,-0.01144,-0.16641,0.01152,0.16652,0,0,0,-0
3976,251252,481,48,1436406120,850,1,0,3,2,ACQUIRING,CROSSING,0,0,-0.01144,-0.16641,0.01152,0.16652,0,0,0,-0
3976,251252,482,16,1436406132,441,1,1,3,2,ACQUIRING,CROSSING,20,-0.06243,-0.01144,-0.19762,0.01152,0.19774,0,-0.031215,0,0.031215
3976,251252,482,20,1436406133,877,1,1,3,2,ACQUIRING,CROSSING,18,-0.06243,-0.01144,-0.19762,0.01152,0.19774,0,-0.031215,0,0.031215
3976,251252,482,24,1436406135,330,1,1,3,2,ACQUIRING,CROSSING,16,-0.06243,-0.01144,-0.19762,0.01152,0.19774,0,-0.031215,0,0.031215
3976,251252,482,28,1436406136,880,1,1,3,2,ACQUIRING,CROSSING,15,-0.06243,-0.01144,-0.19762,0.01152,0.19774,0,-0.031215,0,0.031215
3976,251252,482,32,1436406138,243,1,1,3,2,ACQUIRING,CROSSING,14,-0.06243,-0.01144,-0.19762,0.01152,0.19774,0,-0.031215,0,0.031215
3976,251252,482,36,1436406139,796,1,1,3,2,ACQUIRING,CROSSING,12,-0.06243,-0.01144,-0.19762,0.01152,0.19774,0,-0.031215,0,0.031215
3976,251252,482,40,1436406141,243,1,1,3,2,ACQUIRING,CROSSING,11,-0.06243,-0.01144,-0.19762,0.01152,0.19774,0,-0.031215,0,0.031215
3976,251252,482,44,1436406142,696,1,1,3,2,ACQUIRING,CROSSING,9,-0.06243,-0.01144,-0.19762,0.01152,0.19774,0,-0.031215,0,0.031215
3976,251252,482,48,1436406144,137,1,1,3,2,ACQUIRING,CROSSING,8,-0.06243,-0.01144,-0.19762,0.01152,0.19774,0,-0.031215,0,0.031215
3976,251252,482,52,1436406145,569,1,1,3,2,ACQUIRING,CROSSING,6,-0.06243,-0.01144,-0.19762,0.01152,0.19774,0,-0.031215,0,0.031215
```

Scan dip info

- The data written via dip to the csv file has the format described in https://hypernews.cern.ch/HyperNews/CMS/get/AUX/2014/03/24/12:24:29-79902-LHC-OP-ES-0027-V1.pdf
- This is the meaning of the columns:
  - fillnum,runnum,lsnum,nbnum,tssec,tsmsec,acqflag,step,beam,ip,scanstatus,plane,progress,nominal_separation,read_nominal_B1sepPlane,read_nominal_B1xingPlane,read_nominal_B2sepPlane,read_nominal_B2xingPlane,set_nominal_B1sepPlane,set_nominal_B1xingPlane,set_nominal_B2sepPlane,set_nominal_B2xingPlane
- To find when scan data was acquired for IP5 (=2^5 = 32), do
  - grep '32,AC' vdm_093db38f-7347-4f1a-b551-9f2265852db8.csv
- "nominal_separation" tells you the nominal separation of the two beams, "0" being heads-on
- "plane" has two possible values:
  - SEPARATION: is the vertical plane in CMS, i.e. Y
  - CROSSING : is the horizontal plane in CMS, i.e. X

Information you need to hand-extract from the scan dip file

- The fill number
- The time window of the X and the Y scan at P5
  - in order to identify the luminometer raw data files
  - in order to tell the framework config file what the approximate search window is for the X and the Y scans you want to pair to get a lumi calibration
- Specifically for the scan in fill 3976 on July 9:
  - Inspect the output of
    - grep '32,AC' vdm_093db38f-7347-4f1ab551-9f2265852db8.csv
  - This will show you that there were two optimization scans done in fill 3976, one early on in the fill, on July 8, another towards the end, in the early morning of July 9
  - The time window for the one on July 9 is:
    - [436406667, 1436407238]
  - In human readable:
    - [Thu, 09 Jul 2015 01:51:07 GMT, Thu, 09 Jul 2015 02:00:38 GMT]
  - NB: You can use for the conversion
    - http://www.epochconverter.com
    - or pandas.to_datetime(1436406667)

Where to find your data (III)

- This is a moving target. Below I give he information for fill 3976, scan of July 9:
- We are looking for data files with this timestamp in their name: 150709*,
- On the online cluster :
  - scan dip data in csv :
    - ls -alt /brildata/vdm/vdm*
  - Luminometer data is temporarily here:
    - On srv-s2d16-21-01, scratch area:
    - In our particular case: /scratch/central3

```
a5063350-576f-490b-82a6-2c63cf6279a3_1507090150_256.hd5
a5063350-576f-490b-82a6-2c63cf6279a3_1507090153_257.hd5
a5063350-576f-490b-82a6-2c63cf6279a3_1507090155_258.hd5
a5063350-576f-490b-82a6-2c63cf6279a3_1507090158_259.hd5
a5063350-576f-490b-82a6-2c63cf6279a3_1507090200_260.hd5
```

- To make your live simpler, I've copied all relevant files onto /brildata:

```
[grothe@srv-C2C04-07 brildata]$ pwd
/brildata
[grothe@srv-C2C04-07 brildata]$ ls -l scan*
-rw-r--r-- 1 grothe  zh 36230771 Jul  9 14:27 scanFill3835_150607.tar.gz
-rw-r--r-- 1 grothe  zh 67217360 Jul  9 14:31 scanFill3858_150614.tar.gz
-rw-r--r-- 1 grothe  zh  2843069 Jul  9 10:52 scanFill3960_150704.tar.gz
-rw-r--r-- 1 grothe  zh 41534188 Jul  9 14:38 scanFill3976_150709.tar.gz
```

For older scans: Where to find your data (IV)

- For older scans, for example the one on May 30:
- In the following, if there is a timestamp in the name of the file, it's of the format: _150530*, so for the Sat scans look for files with *_15053018*
- on the online cluster, :
  - vdm dip data in csv :
    - ls -alt /brildata/vdm/vdm*
  - beam data (FBCT, DCCT, beam patterns) during vdm scan in csv
    - ls -alt /brildata/vdm/dipdata_*
  - HF data & beam data (same info as in csv file, but in hdf5 format):
    - ls -alt /brildata/central/*_15053018*.hd5
  - PLT data:
    - ls -alt /brildata/plt/brildaq_output/*_15053018*.hd5
  - BCM1F data:
    - ls -alt /brildata/data_bcm1fprocessor/*_15053018*.hd5

- David copied the relevant files to /brildata/scanFill3804_15053018 on srv-s2d16-21-01

Framework code

- You can look at it on the web:
  - https://svnweb.cern.ch/cern/wsvn/VdMframework/work/Monika/
- To check out:
  - svn co svn+ssh://username@svn.cern.ch/reps/VdMframework/work/Monika vdm

Analysis on lxplus

- Sign out the latest framework code
  - svn co svn+ssh://username@svn.cern.ch/reps/VdMframework/work/Monika vdm
- Set up your environment: For tcsh
  - setenv LD_LIBRARY_PATH /afs/cern.ch/cms/lumi/brilconda-1.0.3/root/lib
  - setenv PYTHONPATH /afs/cern.ch/cms/lumi/brilconda-1.0.3/root/lib
  - setenv PYTHONPATH ${PWD}:$PYTHONPATH
  - setenv PYTHONPATH ${PWD}/fits:$PYTHONPATH
  - setenv PYTHONPATH ${PWD}/corrections:$PYTHONPATH
  - setenv PYTHONPATH $ROOTSYS/lib:$PYTHONPATH
  - setenv VDMPATH ${PWD}
  - setenv PATH /afs/cern.ch/cms/lumi/brilconda-1.0.3/bin:$PATH
- For bash:
  - export LD_LIBRARY_PATH=/afs/cern.ch/cms/lumi/brilconda-1.0.3/root/lib
  - export PYTHONPATH=/afs/cern.ch/cms/lumi/brilconda-1.0.3/root/lib
  - export PYTHONPATH=${PWD}:$PYTHONPATH
  - export PYTHONPATH=${PWD}/fits:$PYTHONPATH
  - export PYTHONPATH=${PWD}/corrections:$PYTHONPATH
  - export PYTHONPATH=$ROOTSYS/lib:$PYTHONPATH
  - export VDMPATH=${PWD}
  - export PATH=/afs/cern.ch/cms/lumi/brilconda-1.0.3/bin:$PATH
- After that do:
  - source /afs/cern.ch/cms/lumi/brilconda-1.0.3/root/bin/thisroot.csh

After checkout from svn:

```
calculateCalibrationConstant_Config.json
calculateCalibrationConstant.py
CorrectionManager.py
corrections
dataPrep_corr
dataPrep_HF
dataPrepII
dataPrep_PCC
dict
env.sh
FitManager.py
fitResultReader.py
fits
forTmpStorage
inputDataReaderII.py
inputDataReader.py
luminometers.py
makeBeamCurrentFile_Config.json
makeBeamCurrentFileII_Config.json
makeBeamCurrentFileII.py
makeBeamCurrentFileII.pyc
makeBeamCurrentFile.py
makeGraphs2D_Config.json
makeGraphs2D.py
makeGraphsFile_Config.json
makeGraphsFileII.py
makeGraphsFileII.py~
makeGraphsFile.py
makeScanFile_Config.json
makeScanFileII_Config_3804.json
makeScanFileII_Config_3829.json
makeScanFileII_Config_3833.json
makeScanFileII_Config_3835.json
makeScanFileII_Config.json
makeScanFileII.py
makeScanFileII.pyc
makeScanFile.py
```

```
vdmDriver_Config.json
vdmDriverII_Config_3804.json
vdmDriverII_Config_BCM1F_3835.json
vdmDriverII_Config_BCM1F_3858.json
vdmDriverII_Config_HF_3835.json
vdmDriverII_Config_HF_3858.json
vdmDriverII_Config_HF_3976_150709.json
vdmDriverII_Config.json
vdmDriverII_Config_PLT_3835.json
vdmDriverII_Config_PLT_3976_150708.json
vdmDriverII_Config_PLT_3976_150709.json
vdmDriverII.py
vdmDriverII.py~
vdmDriver.py
vdmFitter_Config.json
vdmFitterII.py
vdmFitter.py
vdm_runII_test.py
vdmUtilities.py
```

Code specific to runII is marked with “II”

After checkout from svn (II)

- The function of these files is explained in the design document:
  - calculateCalibrationConstant.py
  - makeBeamCurrentFileII.py
  - makeGraphsFileII.py
  - makeScanFileII.py
  - vdmDriverII.py
  - vdmFitterII.py
- In addition:
  - CorrectionManager.py: Plugin manager for corrections
  - FitManager.py: Plugin manager for fits
  - inputDataReader.py: Contains helper class that can be instantiated in the framework code, to hold the input data as read in from the output pickle files of makeScanFileII.py, makeBeamCurrentFileII.py and dataPrepII/makeRateFile.py
  - luminometers.py: Contains helper classes that hold information that characterizes each luminometer, used by calculateCalibrationConstant.py
  - makeGraphs_2D.py: Puts 1D graphs as produced by makeGraphsFile.py together into 2D graphs, not yet used for runII
  - vdmUtilities.py: Contains general tools

After checkout from svn (III)

- The relevant subdirectories for runII:
  - corrections/
    - holds the doer classes that are used by makeGraphsFilesII.py to apply a correction to a given graph
    - for the moment not used in runII
  - fits/
    - holds the fit functions that are used by vdmFitterII.py to fit a given graph
  - dataPrepII
    - This is were the code resides to extract the luminometer rates from the hdf5 files

Running the prototype

Step 1: edit env.sh such that it is appropriate for the setup you're working in

Step 2:

> source env.sh

Step 3: Edit the relevant vdmDriverII config json file, i.e.

vdmDriverII_Config_HF_3976_150709.json

or vdmDriverII_Config_PLT_3976_150709.json

Step 4: For example

> python vmdDriverII.py vdmDriverII_Config_PLT_3976_150709.json

- What you are running and in which order is defined in vdmDriverII_Config_etcetcetc.json
- The json file shown in the next slide is for a full analysis without corrections to the graphs.

vdmDriverII_Config.json

```
"Fill": "3976",
"Date": "July092015",
"Luminometer": "PLT",
"AnalysisDir": "Fill3976_July092015",
"CorrLater" : ["LengthScale", "Ghosts", "Satellites"],
"Corr" : ['noCorr'],
"Comment": "Output of following step goes to <AnalysisDir>/cond",
"makeScanFile": true,
"Comment": "Output of following step goes to <AnalysisDir>/LuminometerData",
"makeRateFile": true,
"Comment": "Output of following step goes to <AnalysisDir>/cond",
"makeBeamCurrentFile": true,
"Comment": "Output of following step goes to <AnalysisDir>/corr",
"makeBeamBeamFile": false,
"Comment": "Output of following step goes to <AnalysisDir>/corr",
"makeGhostsFile": false,
"Comment": "Output of following step goes to <AnalysisDir>/corr",
"makeSatellitesFile": false,
"Comment": "Output of following step goes to <AnalysisDir>/<Luminometer>/graphs",
"makeLengthScaleFile": false,
"Comment": "Output of following step goes to <AnalysisDir>/corr",
"makeGraphsFile": true,
"Comment": "Output of following step goes to <AnalysisDir>/corr",
"makeGraphs2D": false,
"Comment": "Output of following step goes to <AnalysisDir>/<Luminometer>/results/<Corr>",
"runVdmFitter": true,
```

The analysis flow comprises all steps indicated as “true”, i.e., in order: makeScanFile -> makeHFRateFile -> makeBeamCurrentFile -> makeGraphsFile -> runVdmFitter. The output will go into the directory “Fill3976_Jul092015”.

vdmDriverII_Config.json (II)

Each step in the analysis flow needs configuration data, which is also contained in vdmDriver_Config.json, see:

- makeScanFileConfig
- makeHFRateFileConfig
- makeBeamCurrentFileConfig
- makeGraphsFileConfig
- runVdmFitterConfig

The input/output file names of the analysis steps follow the convention defined in the design document. However note that they are all configurable via vdmDriver_Config.json
You have total freedom to modify them to your liking, however you need to take care to follow through with your modifications such that all steps in the analysis can still find their intended input files.

vdmDriverII sets up a default directory structure.
The top level directory of that dir structure has the name specified in the json as “AnalysisDir”.

As specified in the design document, information/data exchange between the analysis steps happens via python pickle files. Where ever possible, the output is also written into csv files. These are human readable and provide a handy tool to look at the information content and flow in the analysis.

vdmDriverII_Config.json (III)

```
"makeScanFileConfig":
        {
        "InputCentralPath": "/Users/grothe/data/VdM/scanFill3976_150709/central3",
        "InputDIPFile" : "/Users/grothe/data/VdM/scanFill3976_150709/vdm/vdm_093db38f-7347-4f1a-b551-9f2265852db8.csv",
        "ScanNames" : ["Y1","X1"],
        "Comment" : "Since X,Y scans are automatically differentiated, a time window can include an X and a Y scan, will still work, timestamp in UTC",
        "ScanTimeWindows" :[[1436406667, 1436407238],[1436406667, 1436407238]],
        "Comment" : "A few parameters that are not in our current DIP file, but should be available eventually",
        "Comment" : "betaStar in cm, angle in microrad",
        "BetaStar" : 80.0,
        "Angle" : 145.0,
        "Offset" : [0.0,0.0],
        "ParticleTypeB1" : "proton",
        "ParticleTypeB2" : "proton",
        "Comment": "Beam energies in GeV",
        "EnergyB1" : 6500,
        "EnergyB2" : 6500,
        "OutputSubDir" : "cond"
        },
```

Defining the scan time windows is essential for running the framework and cannot be extracted automatically from DIP information.

You only need to loosely set the time window such that the X, Y scans that you wish to group together are contained in the window and that they don't overlap with the window of another possible X/Y pairing.

Running the prototype (II)
Output to screen

```
Running with this config info:
Fill  3976
Date  July092015
Luminometer  PLT
AnalysisDir  Fill3976_July092015
Corrections  [u'noCorr']
makeScanFile  True
makeBeamCurrentFile  True
makeGraphsFile  True
runVdmFitter  True

-->> Make directories:
./Fill3976_July092015
./Fill3976_July092015/corr
./Fill3976_July092015/cond
./Fill3976_July092015/LuminometerData
./Fill3976_July092015/PLT
./Fill3976_July092015/PLT/graphs
./Fill3976_July092015/PLT/results
./Fill3976_July092015/PLT/results/noCor

Running makeScanFile with config info:
Comment Beam energies in GeV
ScanTimeWindows [[1436406667, 143640723
InputCentralPath /afs/cern.ch/work/g/gr
```

etc....

etc....

```
-->> Available correction plugins :
BeamBeam_Corr
Satellites_Corr
Ghosts_Corr
LengthScale_Corr
<<--

The following corrections will be applied, in order:
Corr #1: noCorr
Now applying correction:  noCorr
Now at Scan number  1
Now at Scan number  2
Running runVdmFitter with config info:
Comment Input graphs file in <AnalysisDir>/<Luminometer>/graphs
InputGraphsFile graphs/graphs_3976_noCorr.pkl
FitName SGConst
FitConfigFile fits/SGConst_Config.json


ATTENTION: Output will be written into  ./Fill3976_July092015/PLT/results/noCorr/
Please check there for log files.

-->> Available fit plugins :
SG_Fit
DG_Fit
DGConst_Fit
SGConst_Fit
DG_2D_Fit
GSupGConst_Fit
<<--
```

Running the prototype – directories and their contents

The directory structure is set up by vdmDriverII

See next slide

```
[grothe@lxplus0100 vdm]$ ls -l Fill3976_July092015
total 8
drwxr-xr-x. 2 grothe zh 2048 Jul 14 17:11 cond
drwxr-xr-x. 2 grothe zh 2048 Jul 14 17:10 corr
drwxr-xr-x. 2 grothe zh 2048 Jul 14 17:11 LuminometerData
drwxr-xr-x. 4 grothe zh 2048 Jul 14 17:10 PLT
[grothe@lxplus0100 vdm]$ ls -l Fill3976_July092015/LuminometerData/
total 301
-rw-r--r--. 1 grothe zh  94291 Jul 14 17:11 Rates_PLT_3976.csv
-rw-r--r--. 1 grothe zh 212004 Jul 14 17:11 Rates_PLT_3976.pkl
[grothe@lxplus0100 vdm]$ ls -l Fill3976_July092015/cond
total 425
-rw-r--r--. 1 grothe zh 115887 Jul 14 17:11 BeamCurrents_3976.csv
-rw-r--r--. 1 grothe zh 284946 Jul 14 17:11 BeamCurrents_3976.pkl
-rw-r--r--. 1 grothe zh  21834 Jul 14 17:11 checkFBCTcalib_3976.pdf
-rw-r--r--. 1 grothe zh   3342 Jul 14 17:10 Scan_3976.csv
-rw-r--r--. 1 grothe zh   5346 Jul 14 17:10 Scan_3976.pkl
[grothe@lxplus0100 vdm]$ ls -l Fill3976_July092015/corr
total 0
[grothe@lxplus0100 vdm]$ ls -l Fill3976_July092015/PLT
total 4
drwxr-xr-x. 2 grothe zh 2048 Jul 14 17:11 graphs
drwxr-xr-x. 3 grothe zh 2048 Jul 14 17:10 results
[grothe@lxplus0100 vdm]$ ls -l Fill3976_July092015/PLT/graphs
total 362
-rw-r--r--. 1 grothe zh 282019 Jul 14 17:11 graphs_3976_noCorr.pkl
-rw-r--r--. 1 grothe zh  87654 Jul 14 17:11 graphs_3976_noCorr.root
[grothe@lxplus0100 vdm]$ ls -l Fill3976_July092015/PLT/results
total 2
drwxr-xr-x. 2 grothe zh 2048 Jul 14 17:11 noCorr
[grothe@lxplus0100 vdm]$ ls -l Fill3976_July092015/PLT/results/noCorr
total 4065
-rw-r--r--. 1 grothe zh   70367 Jul 14 17:11 SGConst_FitResults.csv
-rw-r--r--. 1 grothe zh   83128 Jul 14 17:11 SGConst_FitResults.pkl
-rw-r--r--. 1 grothe zh  892388 Jul 14 17:11 SGConst_FittedGraphs.pdf
-rw-r--r--. 1 grothe zh 2247543 Jul 14 17:11 SGConst_FittedGraphs.root
-rw-r--r--. 1 grothe zh  260964 Jul 14 17:11 SGConst.log
-rw-r--r--. 1 grothe zh  606041 Jul 14 17:11 SGConst_Minuit.log
```

results of the analysis go into the dir specified as “Luminometer” in the json, here ‘PLT’

PLT rates per scan point and BCID

beam currents per scan point and BCID

time window and displacement per scan point

empty because we specified “Corr” in the json to be ‘noCorr’

graphs are put together from rates, currents, scan point info; label ‘noCorr’ as specified in json

analysis results; ‘noCorr’ as specified in json “Corr”; ‘SGConst’ is the fit function as specified in json “FitName”

Running with a different fit function

Up to now, the analysis was done with a single Gaussian plus constant as fit function.
Configuration for this in vdmDriverII_Config.json

```
        },
    "vdmFitterConfig":
            {
            "Comment": "Input graphs file in <AnalysisDir>/<Luminometer>/graphs",
            "InputGraphsFile" : "graphs/graphs_3976_noCorr.pkl",
            "FitName" : "SGConst",
            "FitConfigFile" : "fits/SGConst_Config.json"
            }
```

To use a double Gaussian plus constant instead:

```
        },
    "vdmFitterConfig":
            {
            "Comment": "Input graphs file in <AnalysisDir>/<Luminometer>/graphs",
            "InputGraphsFile" : "graphs/graphs_3976_noCorr.pkl",
            "FitName" : "DGConst",
            "FitConfigFile" : "fits/DGConst_Config.json"
            }
```

Running with a different fit function (II)

However, just to rerun the analysis with a different fit function doesn't require to redo the graphs. Only the fitting step needs to be redone. Hence all steps other than the fitting step should be set to false in vdmDriverII_Config.json:

```
"Fill": "3976",
"Date": "July092015",
"Luminometer": "PLT",
"AnalysisDir": "Fill3976_July092015",
"CorrLater" : ["LengthScale", "Ghosts", "Satellites"],
"Corr" : ["noCorr"],
"Comment": "Output of following step goes to <AnalysisDir>/cond",
"makeScanFile": false,
"Comment": "Output of following step goes to <AnalysisDir>/LuminometerData",
"makeRateFile": false,
"Comment": "Output of following step goes to <AnalysisDir>/cond",
"makeBeamCurrentFile": false,
"Comment": "Output of following step goes to <AnalysisDir>/corr",
"makeBeamBeamFile": false,
"Comment": "Output of following step goes to <AnalysisDir>/corr",
"makeGhostsFile": false,
"Comment": "Output of following step goes to <AnalysisDir>/corr",
"makeSatellitesFile": false,
"Comment": "Output of following step goes to <AnalysisDir>/<Luminometer>/graphs",
"makeLengthScaleFile": false,
"Comment": "Output of following step goes to <AnalysisDir>/corr",
"makeGraphsFile": false,
"Comment": "Output of following step goes to <AnalysisDir>/corr",
"makeGraphs2D": false,
"Comment": "Output of following step goes to <AnalysisDir>/<Luminometer>/results/<Corr>",
"runVdmFitter": true,
```

Running with a different fit function – Output to screen

```
Running with this config info:
Fill  3976
Date  July092015
Luminometer  PLT
AnalysisDir  Fill3976_July092015
Corrections  [u'noCorr']
makeScanFile  False
makeBeamCurrentFile  False
makeGraphsFile  False
runVdmFitter  True

-->> Make directories:
```

directory structure exists already, hence vdmDriver does not redo it

```
Running runVdmFitter with config info:
Comment Input graphs file in <AnalysisDir>/<Luminometer>/graphs
InputGraphsFile graphs/graphs_3976_noCorr.pkl
FitName DGConst
FitConfigFile fits/DGConst_Config.json


ATTENTION: Output will be written into  ./Fill3976_July092015/PLT/results/noCorr/
Please check there for log files.

-->> Available fit plugins :
SG_Fit
DG_Fit
DGConst_Fit
SGConst_Fit
DG_2D_Fit
GSupGConst_Fit
<<--
```

Running with a different fit function – directories and their content

```
[grothe@lxplus0100 vdm]$ ls -l Fill3976_July092015/PLT/results/noCorr/
total 8565
-rw-r--r--. 1 grothe zh   92526 Jul 14 17:27 DGConst_FitResults.csv
-rw-r--r--. 1 grothe zh  107099 Jul 14 17:27 DGConst_FitResults.pkl
-rw-r--r--. 1 grothe zh  931321 Jul 14 17:28 DGConst_FittedGraphs.pdf
-rw-r--r--. 1 grothe zh 2316589 Jul 14 17:28 DGConst_FittedGraphs.root
-rw-r--r--. 1 grothe zh  327330 Jul 14 17:27 DGConst.log
-rw-r--r--. 1 grothe zh  829582 Jul 14 17:27 DGConst_Minuit.log
-rw-r--r--. 1 grothe zh   70367 Jul 14 17:11 SGConst_FitResults.csv
-rw-r--r--. 1 grothe zh   83128 Jul 14 17:11 SGConst_FitResults.pkl
-rw-r--r--. 1 grothe zh  892388 Jul 14 17:11 SGConst_FittedGraphs.pdf
-rw-r--r--. 1 grothe zh 2247543 Jul 14 17:11 SGConst_FittedGraphs.root
-rw-r--r--. 1 grothe zh  260964 Jul 14 17:11 SGConst.log
-rw-r--r--. 1 grothe zh  606041 Jul 14 17:11 SGConst_Minuit.log
```

- SGConst_FitResults.csv: input files for the calculateCalibration step.
- SGConst_FittedGraphs.pdf: graphs overlaid with the fit function, per bcid and per scan
- SGConst.log is a log file. In case there are errors when running the fit, e.g. bugs in the fitting code, they would show up in this log file.
- SGConst_Minuit.log is the log of the minuit output. To determine whether any of the fits did not converge properly, see below. This means out of 220 fits, 4 failed.

```
[grothe@lxplus0100 vdm]$ grep "STATUS=FAILED" Fill3976_July092015/PLT/results/noCorr/SGConst_Minuit.log
 FCN=230.03 FROM MIGRAD    STATUS=FAILED        209 CALLS         210 TOTAL
 FCN=2621.74 FROM MIGRAD    STATUS=FAILED        207 CALLS         208 TOTAL
 FCN=1954.01 FROM MIGRAD    STATUS=FAILED        191 CALLS         192 TOTAL
 FCN=2546.85 FROM MIGRAD    STATUS=FAILED        209 CALLS         210 TOTAL
```

More on fitting

- All available fit functions are in the directory fits/
- Each fit comes with a config file that sets limits and starting values
- The user can add new fit functions
- In order to add a fit function not yet available in fits/:
  - just add a new myfit_Fit.py and myfit_Config.json in the directory
  - use DGConst as template and modify it as needed
  - the vdmFitterII will pick up the new fit function automatically because of the line
    - import FitManager

- In order to debug your fit, look at the log files in <AnalysisDir>/<Luminometer>/results/<Corr>/, in our example:
    - Fill3563_Feb142013/HF/results/noCorr/SGConst.log
- In order to verify that the fits converged properly, one can also look at the minuit output, in our example:
    - Fill3563_Feb142013/HF/results/noCorr/SGConst_Minuit.log

Fit Config files

- All available fit functions are in the directory fits/
- Each fit comes with a config file that sets limits and starting values
- Note if upper limit set below lower limit, this means limits will be ignored when doing the fit
- Since FitConfigFile is a parameter set in vdmDriverII_Config.json, user can maintain several configs for the same fit function, for example for the purpose of tuning the start values
- The name of the fit config file to use can be selected in vdmDriverII_Config.json

```
new-host-2:prototype_9Apr15 grothe$ more fits/SGConst_Config.json
{
"StartSigma" : 1.0,
"LimitsSigma" : [0.0, -1.0],
"StartPeak" : 1.0,
"LimitsPeak" : [0.0, -1.0],
"StartConst" : 0.0,
"LimitsConst" : [0.0, 0.01]
}
```

For the expert user

- It is possible to run either a full analysis sequence or single steps in the sequence separately via vdmDriverII.py, by suitably modifying vmdDriverII_Config_whatever.json
- For debugging it is occasionally simpler to work directly with the code to run a single step
- This is possible because every step comes with python code that can be run separately and with its own individual config file
  - Example: Redoing the graphs
    - python makeGraphsFileII.py makeGraphsFileII_Config.py
  - Example: Running just the fitting step, to tune ones fit parameters
    - python vdmFitterII.py vdmFitterII_Config.json

Extracting the calibration: Calculating σ_{vis}

> python calculateCalibrationConstant.py calculateCalibrationConstant_Config_PLT_3976_150709.json

This will produce the visble cross section as function of bcid.
In the config file, one specifies which fit results will be used for the calculation.
The output is here:

```
[grothe@lxplus0045 vdm]$ ls -l Fill3976_July092015/PLT/results/noCorr/
total 8583
-rw-r--r--. 1 grothe zh   92526 Jul 14 17:27 DGConst_FitResults.csv
-rw-r--r--. 1 grothe zh  107099 Jul 14 17:27 DGConst_FitResults.pkl
-rw-r--r--. 1 grothe zh  931321 Jul 14 17:28 DGConst_FittedGraphs.pdf
-rw-r--r--. 1 grothe zh 2316589 Jul 14 17:28 DGConst_FittedGraphs.root
-rw-r--r--. 1 grothe zh  327330 Jul 14 17:27 DGConst.log
-rw-r--r--. 1 grothe zh  829582 Jul 14 17:27 DGConst_Minuit.log
-rw-r--r--. 1 grothe zh    7132 Jul 15 16:37 LumiCalibration_PLT_3976.csv
-rw-r--r--. 1 grothe zh   10615 Jul 15 16:37 LumiCalibration_PLT_3976.pkl
-rw-r--r--. 1 grothe zh   70367 Jul 14 17:11 SGConst_FitResults.csv
-rw-r--r--. 1 grothe zh   83128 Jul 14 17:11 SGConst_FitResults.pkl
-rw-r--r--. 1 grothe zh  892388 Jul 14 17:11 SGConst_FittedGraphs.pdf
-rw-r--r--. 1 grothe zh 2247543 Jul 14 17:11 SGConst_FittedGraphs.root
-rw-r--r--. 1 grothe zh  260964 Jul 14 17:11 SGConst.log
-rw-r--r--. 1 grothe zh  606041 Jul 14 17:11 SGConst_Minuit.log
```

Information in the hdf5 raw data files and how it is handled in the framework

Contents of the hdf5 files

- Currently, the hdf5 files contain several pytables, of which the following can be used by the framework:
  - beam
    - is written into the hdf5 file only when LHC status is either flattop or squeeze or stable beams
    - used in makeBeamCurrentsFile.py
    - contains in particular
      - FBCT current data
      - DCCT current data
      - info on filled bunches in beam1 and beam2
      - info on colliding bunches
  - bcm1flumi: lumi measurement from BCM1F
  - hflumi: lumi measurement from zero counting in HF
  - pltlumi: lumi measurement from counting tracks in the PLT
  - pltlumizero: lumi measurement from counting zeros in the PLT
- All lumi tables (BCM1F, HF, PLT) have the same structure, see next slide:
  - bxraw is the uncalibrated luminosity per 4 nibbles and per bcid
  - avgraw is the uncalibrated luminosity per 4 nibbles, integrated over all bcid
- The lumi tables are used by dataPrepII/makeRateFile.py
  - which table is used as source of the lumi data can be selected in the json config file via the field “RateTable”

Contents of the hdf5 files (II)

```
/pltlumi (Table(105,)) 'pltlumi'
  description := {
  "fillnum": UInt32Col(shape=(), dflt=0, pos=0),
  "runnum": UInt32Col(shape=(), dflt=0, pos=1),
  "lsnum": UInt32Col(shape=(), dflt=0, pos=2),
  "nbnum": UInt32Col(shape=(), dflt=0, pos=3),
  "timestampsec": UInt32Col(shape=(), dflt=0, pos=4),
  "timestampmsec": UInt32Col(shape=(), dflt=0, pos=5),
  "totsize": UInt32Col(shape=(), dflt=0, pos=6),
  "publishnnb": UInt8Col(shape=(), dflt=0, pos=7),
  "datasourceid": UInt8Col(shape=(), dflt=0, pos=8),
  "algoid": UInt8Col(shape=(), dflt=0, pos=9),
  "channelid": UInt8Col(shape=(), dflt=0, pos=10),
  "payloadtype": UInt8Col(shape=(), dflt=0, pos=11),
  "calibtag": StringCol(itemsize=32, shape=(), dflt='', pos=12),
  "avgraw": Float32Col(shape=(), dflt=0.0, pos=13),
  "avg": Float32Col(shape=(), dflt=0.0, pos=14),
  "bxraw": Float32Col(shape=(3564,), dflt=0.0, pos=15),
  "bx": Float32Col(shape=(3564,), dflt=0.0, pos=16),
  "maskhigh": UInt32Col(shape=(), dflt=0, pos=17),
  "masklow": UInt32Col(shape=(), dflt=0, pos=18)}
  byteorder := 'little'
  chunkshape := (102,)
```

- In the <FitFunction>_FittedGraphs.pdf files in, for example, the directory Fill3976_July092015/PLT/results/noCorr/
  - the graphs derived from bxraw are marked with the bcid number to which they belong
  - the graphs derived from avgraw are marked as “sum”