Getting started with BGI simulation

Getting started with BGI simulation

- Virtual-IPM documentation

Installation of virtual-IPM on you local machine

- Find your preferred method from the Virtual-IPM documentation.
- I recommend installing the GUI, it is very useful for creating simulation configuration files.

Installation on lxplus

- Required only if using condor.
- You do not need the GUI there.
- My preferred method:
  - Create a python virtual environment `python3 -m venv myVIPM-env`
  - Activate the Venv `source myVIPM-env/bin/activate`
  - Install virtual-IPM `pip install Virtual-IPM`

General use for virtual-IPM

Create simulation configuration file

- Activate the virtual environment where virtual-ipm is installed, e.g. `conda activate vipm-env` or `source vipm-env/bin/activate`
- Start virtual-IPM GUI `virtual-ipm-gui`.
- Go in each tabs from left to right
  - In **beam**, add a new beam and fill the required informations.
  - In **device**, select *interpolatingIPM* and enter the limit s of the simulation volume. e.g. X -60,60 mm and Y -30, 30 mm
    - X is transverse horizontal, Y is transverse vertical.
    - 0, 0 is the center of the instrument.
  - In **ParticleGeneration**:
    - Select *ZspreadVoitkivDDCS* for a realistic simulation.
    - Select *spreadZeroMomentum* for a realistic electron generation but with electrons that have no initial momentum.
    - In *Zrange* select the length over which electrons are generated. e.g. -10, 10 mm. Typically this is equal or larger than you detection region.

    - In **ParticleTracking**, select *Boris*.
    - In **ElectricGuidingField**:
        - Select *ElectricCSVAdaptor*, if importing an electric field map from CST.
        - Select *UniformElectricField*, if you want a perfect field.
    - In **MagneticGuidingField**, same as for electric field.
    - In **SimulationParameters**,
        - Number of particle = number of electrons/ion that will be generated by the simulation.
        - Particle type
            - *Charge number* = The charge of the particle generated by the simulation.
                - *-1* for electrons.
            - *RestMass* or *RestEnergy* = The rest mass of energy of the particles.
                - Use *RestEnergy* and *%(electron mass energy equivalent in MeV)* for a single electrons.
        - Time Range:
            - Simulation time: The total simulation time. Select something larger than your desired simulation. (Usually the time for a full bunch to travel the BGI volume + electrons drift time.)
            - TimeDelta: The time delta between each simulation steps.
                - I usually use 0.0005 ns (500 fs) but it's probably way overkill.
    - In **Output**:
        - Select basic recorder and choose a name and location for the result file.
- You can open and existing configuration or save it using the first and second icon at the top of the virtual-ipm gui.

Run a simulation on local machine

1. Create a simulation configuration file (.XML).
    - The first one is easily done in the GUI.
    - If you only need to change a single parameter, do it directly in the xml file (don't forget to change the result file name otherwise you will override existing results).
2. Run from the GUI.
    - In the GUI, load the xml file and click on the *Run* button at the top of the GUI.
    - The GUI always runs in verbose mode and is therefore significantly slower than command line mode.
3. Run from the command line.

- With verbose (slow option): `virtual-ipm my_config_file.xml`
- No verbose (fast): `virtual-ipm my_config_file.xml --console-log-level warning` or `--quiet-console`
- Run multiple simulation in parallel on your local machine.
  - Get *GNU parallel* (google it).
  - Make a .txt file with the name of each simulation file. Have only one name per line.
  - Run: `nohup parallel -j 8 -a MYfilename.txt nohup virtual-ipm --quiet-console`
    - the first nohup means that the terminal will not wait on the parallel command to finish. This is optional.
    - The option `-j 8` means that 8 process can run in parallel. You can use as much/little as you want here. I find it best to give it half of my machines physical cores.
    - The second nohup also means not to wait on the virtual-ipm command. (not optional)
    - `--quiet-console` means that virtual-ipm will not prompt anything in the command line.
  - Myfilename.txt is a txt file with the location of one config file per line.

Run a simulation on the grid (condor)

Prerequisite

- You need access to lxplus and cernbox.
- Install virtual-ipm on your lxplus account in its own virtual environment.
- It's good to test that your install is working by using a simple simulation file.

Simulation preparations

This is my way of doing it, not an absolute truth, feel free to change things to your own liking.
On your local machine, in a location synchronised with cernbox

1. Create a top directory with the name you like.
   - This directory must have tree sub-directories, *condor_job*, *files* and *seed*.
2. In *seed*, place the seed configuration file you plan to use to generate the rest of you simulation files.
3. Open virtual-ipm GUI and click on the top right button "parameter sweep".
   - This is a tool used to generate a series of configuration files from a seed file (parameter sweep), where one or several parameters are changed (e.g. beam energy from 1 to 10 TeV in 10 steps).
   - Load the seed file when prompted.
   - "Tick" the parameters that will be swept.
   - Choose the start, stop and step parameters for each.

- Note that the units are fixed by the seed file.
- When done, generate the simulation files by clicking on the *generate* button at the bottom.
- For the location select the *files* folder you created earlier.
- For the result file path set it to blank (delete the default present in the textbox).
- The simulation files should be present under *configurations* and a few more files should have been created.

4. Add a folder called *Results* under *files*. (this is where your simulation results will copied.)
5. In *condor_job*, create three directories: *error*, *log*, *output*.
6. Add the *job.sub* and *run_simulation_job.sh* files. (copy them from an existing project).
7. Add a text file containing the names of each simulation file created earlier (one file per line, file name only, no path).
8. In *job.sub*
   - Change `transfer_input_files` to the EOS location of the *configuration* folder created earlier.
   - Change the `job_flavour` if needed.
   - On the last line, edit the filename `queue config_file from my_list_of_config_files.txt` with the name of the file with each config files name.
9. In *run_simulation_job.sh*:
   - Edit `export PYTHONUSERBASE="/afs/cern.ch/user/s/slevasse/python_for_condor/venv"` to your virtual environment with the install of virtual-ipm.
   - Edit `source /afs/cern.ch/user/s/slevasse/python_for_condor/venv/bin/activate` to match the rigth virtual environment.
   - Edit `output_directory="/eos/project/b/bgi/HL-LHC/simulation/B_field_homogeneity/B_field_homogeneity_04T/files/results"` to match the *result* directory made earlier (using the EOS path).
10. Copy the *condor_job* directory to lxplus.
11. Make sure all the files on your local machine a synchronised with cernbox before to continue further (go get a coffee or something).
12. In lxplus:
    - run the job: `condor_submit job.sub`
    - (Optional) Monitor the job.
      - `condor_wait -status log/NAME_OF_LOG_FILE.log`
      - or `condor_q`
    - If you screwed up and realise it a bit late, Kill the job running on condor.

- `condor_rm jobID`
- jobid is given when running condor_q

13. Results should slowly pile in you *results* folder.