Electron cloud meeting #71, 29/11/2019 (indico)

Participants: E. Buratin, M. Himmerlich, G. Iadarola, K. Li, L. Mether, E. Metral, V. Petit, K. Paraschou, G. Rumolo, L. Sabato, F. Yaman,

Migration of simulation tools to python 3 (G. Iadarola, L. Mether)

Gianni presented some information on the migration of simulation tools to Python 3:

- PyECLOUD, PyHEADTAIL, PyPIC, PyPARIS, etc... have been developed in Python 2.7 as Python 3 and related libraries were not mature enough at that time.
- Python 2.7 will be retired on 31 Dec 2019 (end of official support)
- Python 2 and Python 3 are not compatible with each other. Code changes are required to migrate from py2 to py3. Python 2 and Python 3 libraries cannot work together (e.g. we could not have PyPARIS in py3 and PyEC in py2)
- Therefore we need to migrate all the tools at the same time!
- There are different ways to get python 3 for e-cloud simulations:
    - The simplest way is to install miniconda3 (instructions are available in the PyECLOUD wiki pages);
    - For the clusters (in particular to use MPI) it is more convenient to compile python from the source code (instructions available in the PyECLOUD wiki pages);
    - A public python 3 installation for the team is also available in lxplus.
- Python 3 versions for the entire PyECLOUD-PyHEADTAIL ecosystem have been prepared and tested over the last months The strategy consisted in modifying the Python 2 code, so that the Python 3 code could be automatically generated using the 2to3 tool. This allowed avoiding the development of two codes in parallel.
    - The same strategy will be applied next year to machine follow-up tools.
- Python 3 versions have been released for all packages.
- We will have a transition period (Dec 2019 – Jan 2020), during which:
    - The Python 2.7 versions will remains the default (used for production studies)
    - Pilot studies with Python 3 will be performed on different systems to identify problems that escaped tests done so far.
    - Special procedure to install python 3 versions during the transition period is available on the PyECLOUD wiki.
- At the end of the transition period (mid Jan 2020, if no surprises) the Python 3 versions will become the default.
- First tests on HTCondor revealed no surprises. Further tests will be made on CNAF cluster and on the HPC cluster at CERN.
- Users should take care of migrating their own scripts. Some tips on how to do this are provided in the slides. Gianni and Lotta are available to help.

Multispecies simulations with cross-ionization: investigation on numerical runaway (L. Mether)

Lotta presented an update on electron-ion simulations including cross-ionization effects.

- Cross-ionization starts to have a significant impact as of gas densities of $10^{20}\,m^{-3}$
- Strange behaviors were observed for the simulations with high gas density, in particular sudden drops of the electron density and an unexpected self-sustaining behavior after the bunch train. Investigations were performed to understand if these are physical phenomena or numerical artefacts.
- A first consistency check consisted in monitoring the total energy in the system. This required saving the electric energy of the clouds (previously not implemented in PyECLOUD) together with the kinetic energy.
- After the last bunch passage no external energy is injected into the system, therefore the total energy should stay constant or decrease. Instead, an increase is observed in every energy component, which shows that the evolution after the bunch train is unphysical.
- An energy increase is observed also between bunch passages, showing that the evolution becomes unphysical already during the bunch train passage, and not only during the gap between bunch trains.
- To investigate the unphysical evolution, detailed studies were performed (both time-consuming and data-intensive):
    - The full simulation state was saved at regular intervals during the simulation.
    - Simulations were restarted from saved states and could be run interactively in small steps, allowing the examination of different quantities.
- There appears to be a correlation between the artificial energy growth, the (quasi-)neutralization of the system and increasingly noisy electric fields.
- It looks like the simulation suffers from numerical heating (a well-known problem in plasma simulations), where local noise in the electric field artificially accelerates the electrons, leading to an unphysical energy growth.
- The simulation behavior looks to be affected by macroparticle regenerations. In PyECLOUD simulations, in order to keep the number of macro-particles manageable during the simulation (with exponentially increasing number of particles), macro-particles are suppressed and the other rescaled to keep their number within requested limits When this is done in large steps, a correlation is observed between the regeneration events and the slope of the unphysical energy growth. This shows that the regenerations inject more noise into the system. With a less aggressive regeneration scheme, no evident correlation occurs. The unphysical evolution is slightly damped and postponed, but not fully mitigated.

- Another source of artificial imbalance is introduced in the system by the cross-ionization. Since a probabilistic algorithm is used, which depends on the macro-particle weight of each cloud, small fluctuations in the numbers occur. The introduced imbalance is small compared to the number of particles in the clouds but not so small with respect to the net charge (electrons - ions), which essentially determines the fields. An option was implemented to enforce the algorithm to use the same probabilities for all clouds, removing the imbalance, but this didn't cure the problem.
- To decrease the numerical noise, the number of macro-particles needs to be increased. For these studies the number of macro-particles was pushed up to 8 million per cloud. In order to keep the computational time within reasonable limits a parallelization scheme had to be introduced in the build-up simulation (not available up to now).
- The number of macro-particles per cloud was scanned using the new parallel mode. The numerical breakdown could be pushed to later in the bunch train with increasing macro-particle numbers. Before the breakdown, the simulation is converged w.r.t. the macro-particle number.
- Other parameters were also scanned showing good convergence before the numerical breakdown.
- Increasing further the gas density, the breakdown becomes unavoidable.
- To improve the simulation further, neutrals would need to be tracked and collisions modelled, going to full-scale plasma simulations (would require significant development efforts).
- It is important to note that the simulation is converged w.r.t. numerical parameters until the runaway starts. This indicates that this model is sufficient to cover the evolution until that point
- Based on what we have so far, it is possible to conclude that:
    - Cross-ionization increases the electron and ion densities up to at least 10^{17} m^{-3}.
    - Based on previous instability simulation studies (with a different initial state without cross-ionization, but similar average electron and ion densities), this could be compatible with the observed instabilities.
    - Refined instability simulations could be done starting from the results of build-up simulations with cross-ionization with a reasonable effort (a few more weeks of work).

Update on development for single-particle tracking with e-cloud (K. Paraschou)

Kostas presented an update on the development of a method for single-particle tracking in the presence of e-cloud (already introduced in a previous meeting):

- Macroparticle noise can be significantly reduced by averaging many (2000) electron cloud simulations.

- A closer look to the field interpolation reveals irregularities. An investigation conducted on a 1D analytical case showed that, when using the exact values of the derivative, the error is significantly reduced.
- To improve the accuracy of the derivatives in the case of the e-cloud (for which the analytical values are not available) a refinement procedure was defined:
    - A finer auxiliary grid is introduced;
    - The potential on the boundary nodes of the finer grid is obtained by linear interpolation;
    - The charge density on all nodes of the finer grid is obtained by linear interpolation;
    - Poisson's equation is solved on the finer grid;
    - Finite differences on the finer grid are used to calculate derivatives;
    - The refined potential and derivatives are kept only on the nodes of the original grid.
- This method provides a better interpolation without sacrificing too much memory.
- The grid in PyECLOUD already pushes the limits of a typical RAM. Significant development was required to optimize memory consumption during the refinement procedure. Even then, the solution of Poisson's equation on such a fine grid can easily exceed 100 GBs of memory. For the calculation special resources were used (LIUPSGPU machine of ABP with 24 cores and 256 GB memory, and BigMem nodes in HTCondor).
- Irregularities are significantly suppressed through this procedure. The irregularities were quantified by comparing he symplectic interpolation from the potential against the accurate interpolation on the field. A reduction of the error by more than one order of magnitude is observed when using the refinement procedure.
- Simplified tracking experiments have shown that reducing the irregularities has a significant impact on the beam particle motion.
- The next steps ahead are:
    - Benchmark the interpolation scheme on analytic z-dependent Hamiltonians (RF-multipoles).
    - Check behaviour of "long-term" observables, e.g. Dynamic Aperture.
    - Install e-clouds in the arcs of the LHC model (starting from MAD-X model) and study losses and emittance evolution at injection energy.

Investigation on convergence properties: single pinch (L. Sabato)

Luca presented a follow-up on the convergence properties for PyECLOUD-PyHEADTAIL simulations.

- The lack of convergence for small number of longitudinal slices is visible also on the motion of the individual electrons.
- For a too small number of slices the electrons tend to escape from the bunch, resulting in a smaller density within the bunch. This explains why the bunch results artificially more stable.

- These features are observed for different longitudinal profiles of the bunch and both with and without the externally applied quadrupolar magnetic field.