CERN

Checkpointing for PyECLOUD buildup simulations

Eric Wulff

Acknowledgements: Thanks to Giovanni Iadarola for helpful discussions and input

Outline

- Goal
- How does it work?
- Comparison with previous simulations
- Summary

Goal

- To implement a feature in PyECLOUD making it possible to save checkpoints of the simulation at regular time intervals
- Make the simulation automatically start from the saved checkpoint if it exists

Outline

- Goal
- How does it work?
- Comparison with previous simulations
- Summary

How does it work?

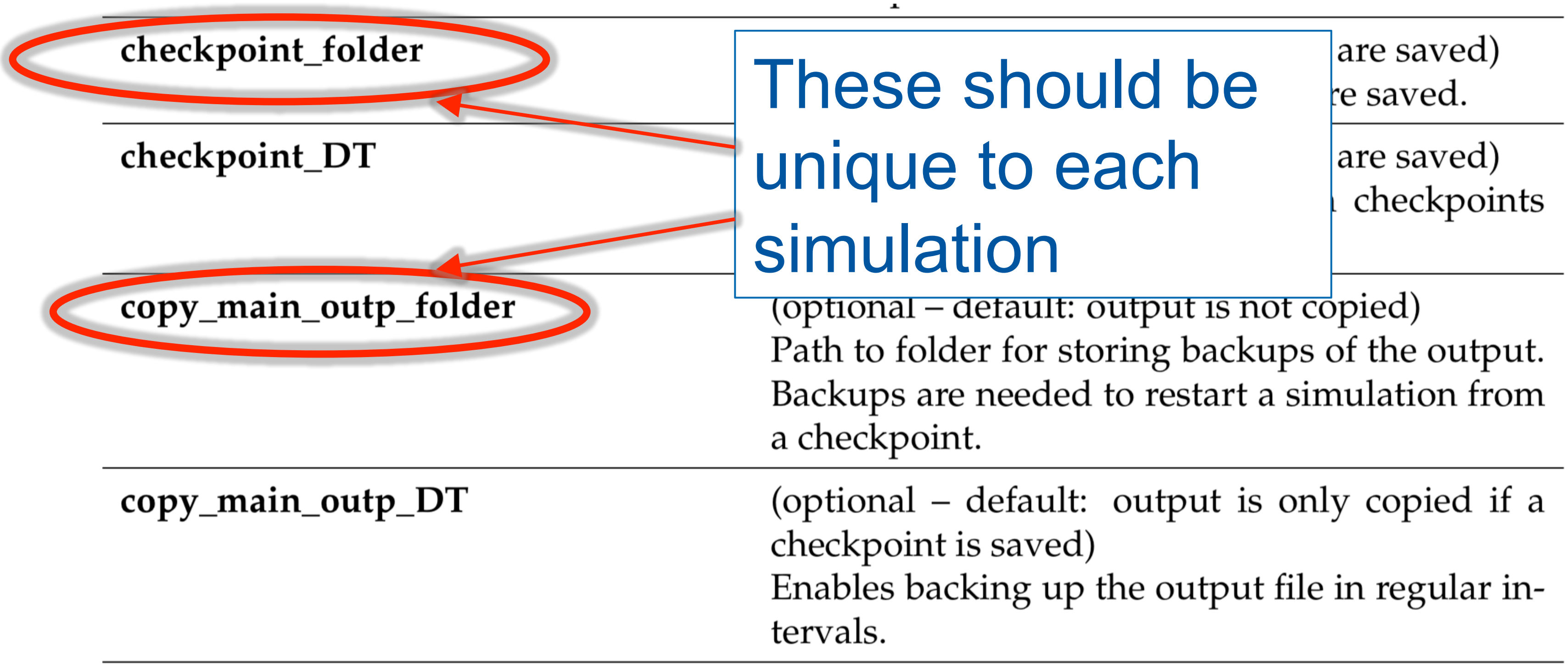
- The checkpointing feature is included with version 7.6.0 of PyECLOUD

- To start a simulation from a checkpoint two things are needed:

  1. A simulation state containing a snapshot of all simulation parameters
  2. A file containing the history of the simulation up until the snapshot (the usual PyECLOUD output file)

How does it work?

- Four new simulation input parameters

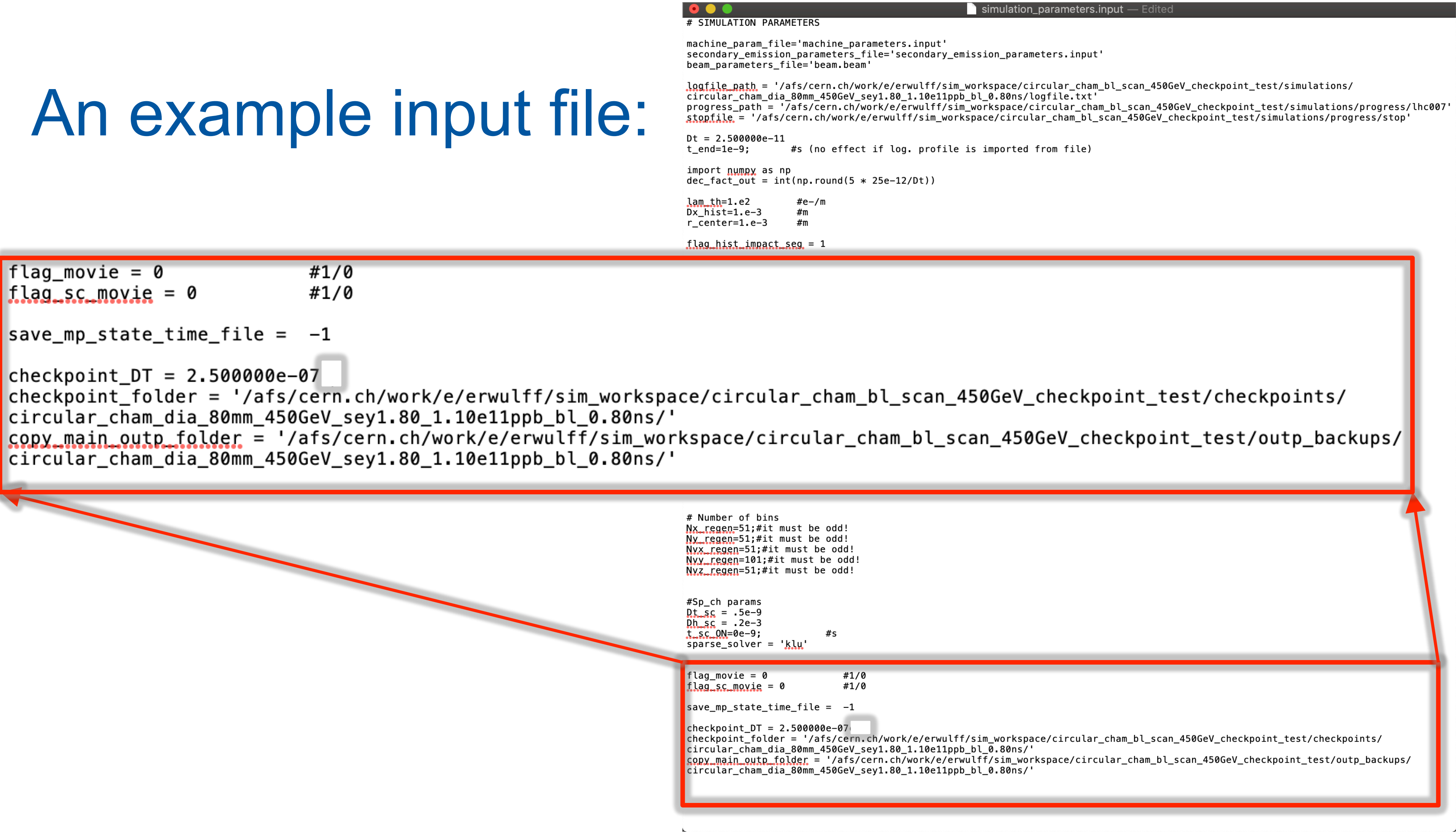


checkpoint_folder	are saved) re saved.
checkpoint_DT	are saved) checkpoints
copy_main_outp_folder	(optional – default: output is not copied) Path to folder for storing backups of the output. Backups are needed to restart a simulation from a checkpoint.
copy_main_outp_DT	(optional – default: output is only copied if a checkpoint is saved) Enables backing up the output file in regular intervals.

These should be unique to each simulation

How does it work?

An example input file:



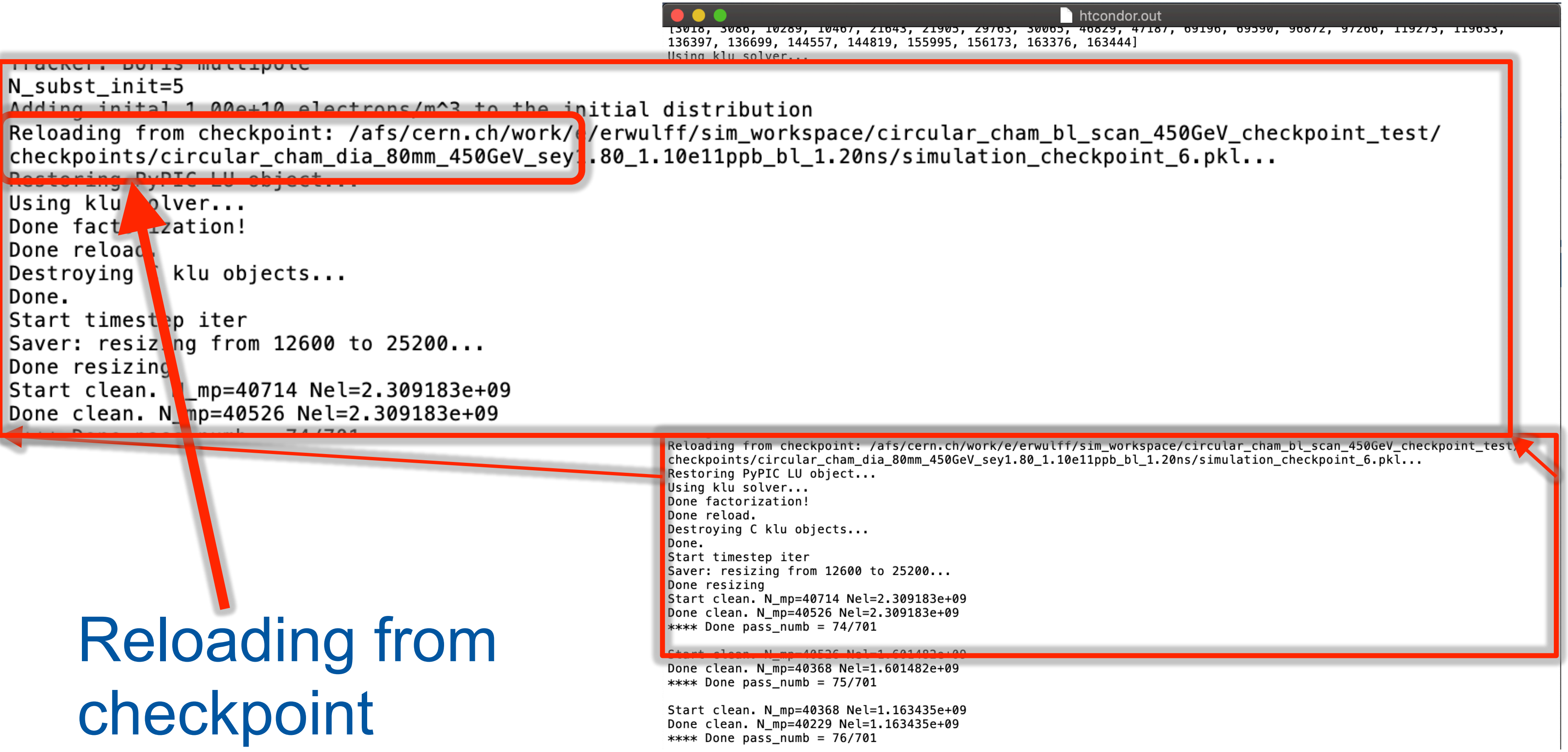
```
# SIMULATION PARAMETERS

machine_param_file='machine_parameters.input'
secondary_emission_parameters_file='secondary_emission_parameters.input'
beam_parameters_file='beam.beam'

logfile_path = '/afs/cern.ch/work/e/erwulff/sim_workspace/circular_cham_bl_scan_450GeV_checkpoint_test/simulations/
circular_cham_dia_80mm_450GeV_sey1.80_1.10e11ppb_bl_0.80ns/logfile.txt'
progress_path = '/afs/cern.ch/work/e/erwulff/sim_workspace/circular_cham_bl_scan_450GeV_checkpoint_test/simulations/progress/lhc007'
stopfile = '/afs/cern.ch/work/e/erwulff/sim_workspace/circular_cham_bl_scan_450GeV_checkpoint_test/simulations/progress/stop'

Dt = 2.500000e-11
t_end=1e-9;       #s (no effect if log. profile is imported from file)

import numpy as np
dec_fact_out = int(np.round(5 * 25e-12/Dt))

lam_th=1.e2       #e-/m
Dx_hist=1.e-3     #m
r_center=1.e-3    #m

flag_hist_impact_seg = 1

# Number of bins
Nx_regen=51;#it must be odd!
Ny_regen=51;#it must be odd!
Nvx_regen=51;#it must be odd!
Nvy_regen=101;#it must be odd!
Nvz_regen=51;#it must be odd!


#Sp_ch params
Dt_sc = .5e-9
Dh_sc = .2e-3
t_sc_ON=0e-9;          #s
sparse_solver = 'klu'

flag_movie = 0            #1/0
flag_sc_movie = 0         #1/0

save_mp_state_time_file =  -1

checkpoint_DT = 2.500000e-07
checkpoint_folder = '/afs/cern.ch/work/e/erwulff/sim_workspace/circular_cham_bl_scan_450GeV_checkpoint_test/checkpoints/
circular_cham_dia_80mm_450GeV_sey1.80_1.10e11ppb_bl_0.80ns/'
copy_main_outp_folder = '/afs/cern.ch/work/e/erwulff/sim_workspace/circular_cham_bl_scan_450GeV_checkpoint_test/outp_backups/
circular_cham_dia_80mm_450GeV_sey1.80_1.10e11ppb_bl_0.80ns/'
```

```
flag_movie = 0            #1/0
flag_sc_movie = 0         #1/0

save_mp_state_time_file =  -1

checkpoint_DT = 2.500000e-07
checkpoint_folder = '/afs/cern.ch/work/e/erwulff/sim_workspace/circular_cham_bl_scan_450GeV_checkpoint_test/checkpoints/
circular_cham_dia_80mm_450GeV_sey1.80_1.10e11ppb_bl_0.80ns/'
copy_main_outp_folder = '/afs/cern.ch/work/e/erwulff/sim_workspace/circular_cham_bl_scan_450GeV_checkpoint_test/outp_backups/
circular_cham_dia_80mm_450GeV_sey1.80_1.10e11ppb_bl_0.80ns/'
```

How does it work?

- The code always checks for a saved checkpoint to restart from if checkpoint_DT and checkpoint_folder are specified
- If checkpoint_DT is specified but not checkpoint_folder the simulation wont run
- copy_main_outp_folder does not have to be specified for the simulation to run, in case it isn’t the code will look for an output file in the local folder

How does it work?

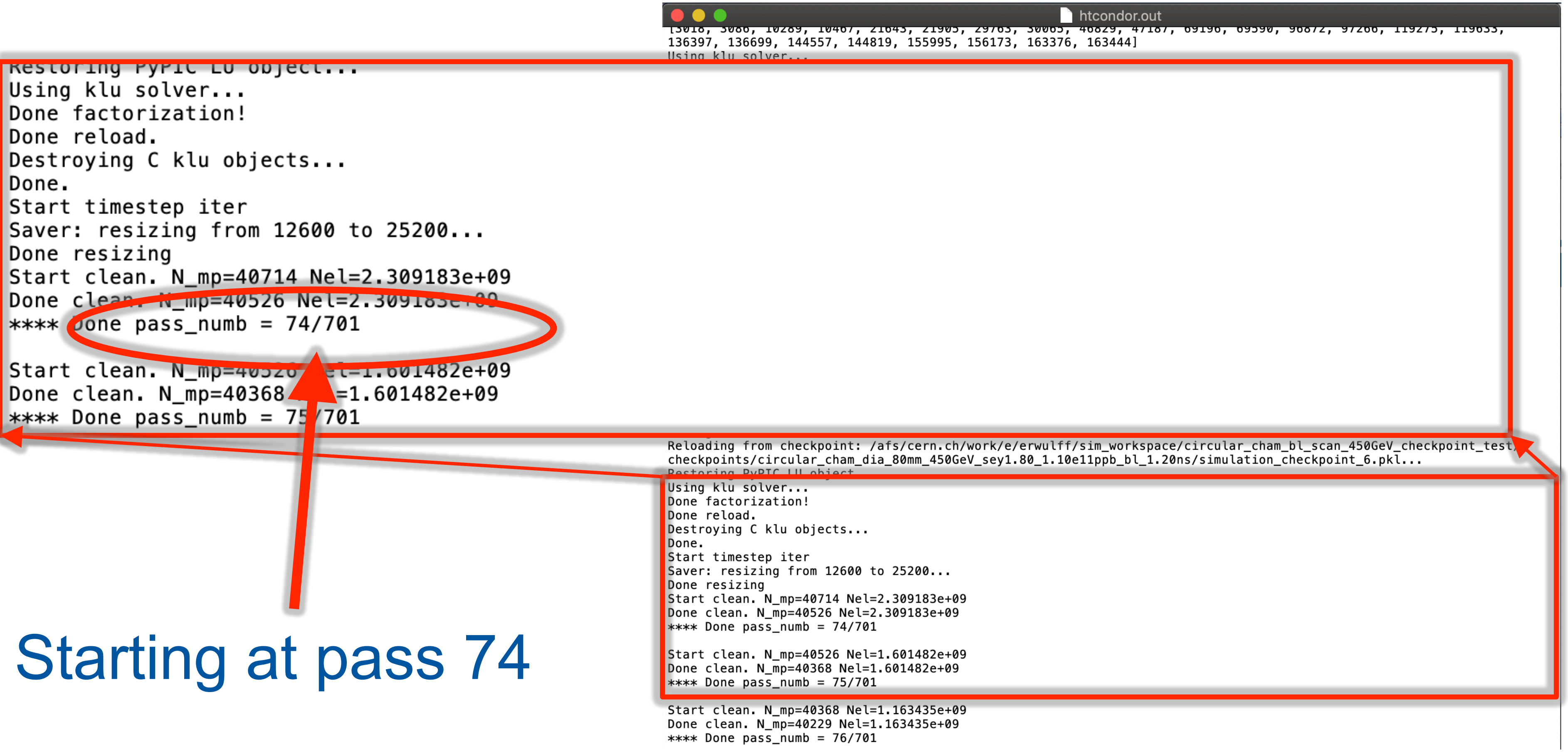


```
Tracker: Boris multipole
N_subst_init=5
Adding inital 1.00e+10 electrons/m^3 to the initial distribution
Reloading from checkpoint: /afs/cern.ch/work/e/erwulff/sim_workspace/circular_cham_bl_scan_450GeV_checkpoint_test/
checkpoints/circular_cham_dia_80mm_450GeV_sey1.80_1.10e11ppb_bl_1.20ns/simulation_checkpoint_6.pkl...
Restoring PyPIC LU object...
Using klu solver...
Done factorization!
Done reload.
Destroying C klu objects...
Done.
Start timestep iter
Saver: resizing from 12600 to 25200...
Done resizing
Start clean. N_mp=40714 Nel=2.309183e+09
Done clean. N_mp=40526 Nel=2.309183e+09
**** Done pass_numb = 74/701
```

```
[3018, 3086, 10289, 10467, 21643, 21905, 29763, 30065, 46829, 47187, 69196, 69590, 96872, 97266, 119275, 119633,
136397, 136699, 144557, 144819, 155995, 156173, 163376, 163444]
Using klu solver...
Reloading from checkpoint: /afs/cern.ch/work/e/erwulff/sim_workspace/circular_cham_bl_scan_450GeV_checkpoint_test/
checkpoints/circular_cham_dia_80mm_450GeV_sey1.80_1.10e11ppb_bl_1.20ns/simulation_checkpoint_6.pkl...
Restoring PyPIC LU object...
Using klu solver...
Done factorization!
Done reload.
Destroying C klu objects...
Done.
Start timestep iter
Saver: resizing from 12600 to 25200...
Done resizing
Start clean. N_mp=40714 Nel=2.309183e+09
Done clean. N_mp=40526 Nel=2.309183e+09
**** Done pass_numb = 74/701

Start clean. N_mp=40526 Nel=1.601482e+09
Done clean. N_mp=40368 Nel=1.601482e+09
**** Done pass_numb = 75/701

Start clean. N_mp=40368 Nel=1.163435e+09
Done clean. N_mp=40229 Nel=1.163435e+09
**** Done pass_numb = 76/701
```

Reloading from checkpoint

How does it work?



```
[3018, 3086, 10289, 10467, 21643, 21905, 29763, 30065, 46829, 47187, 69196, 69590, 96872, 97266, 119275, 119633,
136397, 136699, 144557, 144819, 155995, 156173, 163376, 163444]
Using klu solver...
Reloading from checkpoint: /afs/cern.ch/work/e/erwulff/sim_workspace/circular_cham_bl_scan_450GeV_checkpoint_test/
checkpoints/circular_cham_dia_80mm_450GeV_sey1.80_1.10e11ppb_bl_1.20ns/simulation_checkpoint_6.pkl...
Restoring PyPIC LU object...
Using klu solver...
Done factorization!
Done reload.
Destroying C klu objects...
Done.
Start timestep iter
Saver: resizing from 12600 to 25200...
Done resizing
Start clean. N_mp=40714 Nel=2.309183e+09
Done clean. N_mp=40526 Nel=2.309183e+09
**** Done pass_numb = 74/701

Start clean. N_mp=40526 Nel=1.601482e+09
Done clean. N_mp=40368 Nel=1.601482e+09
**** Done pass_numb = 75/701

Start clean. N_mp=40368 Nel=1.163435e+09
Done clean. N_mp=40229 Nel=1.163435e+09
**** Done pass_numb = 76/701
```

```
Restoring PyPIC LU object...
Using klu solver...
Done factorization!
Done reload.
Destroying C klu objects...
Done.
Start timestep iter
Saver: resizing from 12600 to 25200...
Done resizing
Start clean. N_mp=40714 Nel=2.309183e+09
Done clean. N_mp=40526 Nel=2.309183e+09
**** Done pass_numb = 74/701

Start clean. N_mp=40526 Nel=1.601482e+09
Done clean. N_mp=40368 Nel=1.601482e+09
**** Done pass_numb = 75/701
```

Starting at pass 74

How does it work?

Something to keep in mind:

- After saving a new checkpoint the previous checkpoint is automatically deleted
- If the simulation crashes before it has time to delete the previous checkpoint there will be two saved checkpoints
- This results in the code not knowing which checkpoint to restart from
- It is then up to the user to decide which checkpoint is best to use

Outline

- Goal
- How does it work?
- Comparison with previous simulations
- Summary

Comparison with previous simulations

e-cloud simulation setup of previously done simulations:

- 450 GeV beam energy
- 1.1 ×10^{11} p/bunch beam intensity
- Standard 25 ns beam
- No magnetic field
- Uniform initial electron density
- SEY scan: 1.0 - 1.8
- Bunch length scan: 0.7 - 1.8 ns
- Circular chamber with radius of 40mm

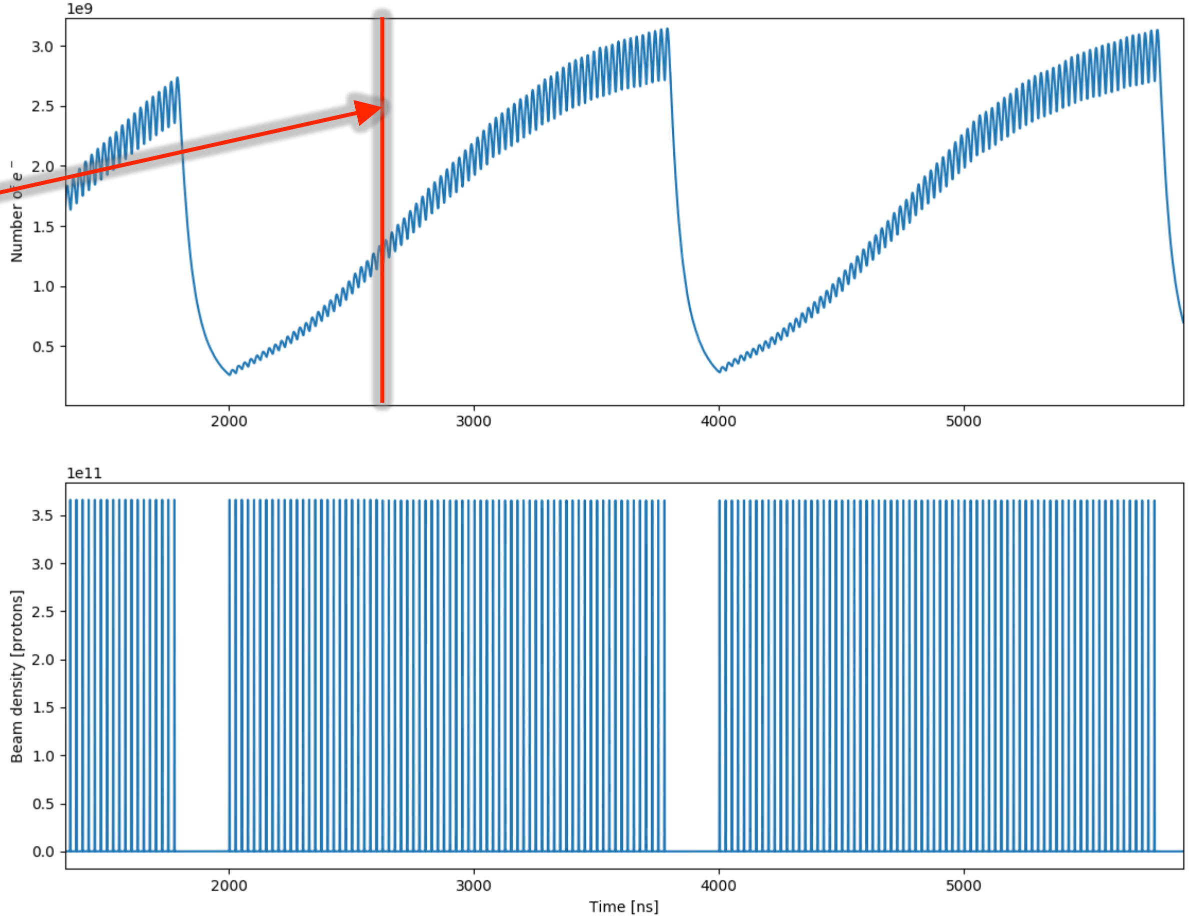
Presented on electron cloud meeting #62

Comparison with previous simulations

New simulations:

- Same simulation parameters as before except scanning fewer parameter values
- SEY values: 1.2, 1.5, 1.8
- Bunch lengths: 0.8, 1.2, 1.6
- These 9 new simulations were run on HTCondor
- All nine simulations were killed and restarted at some point

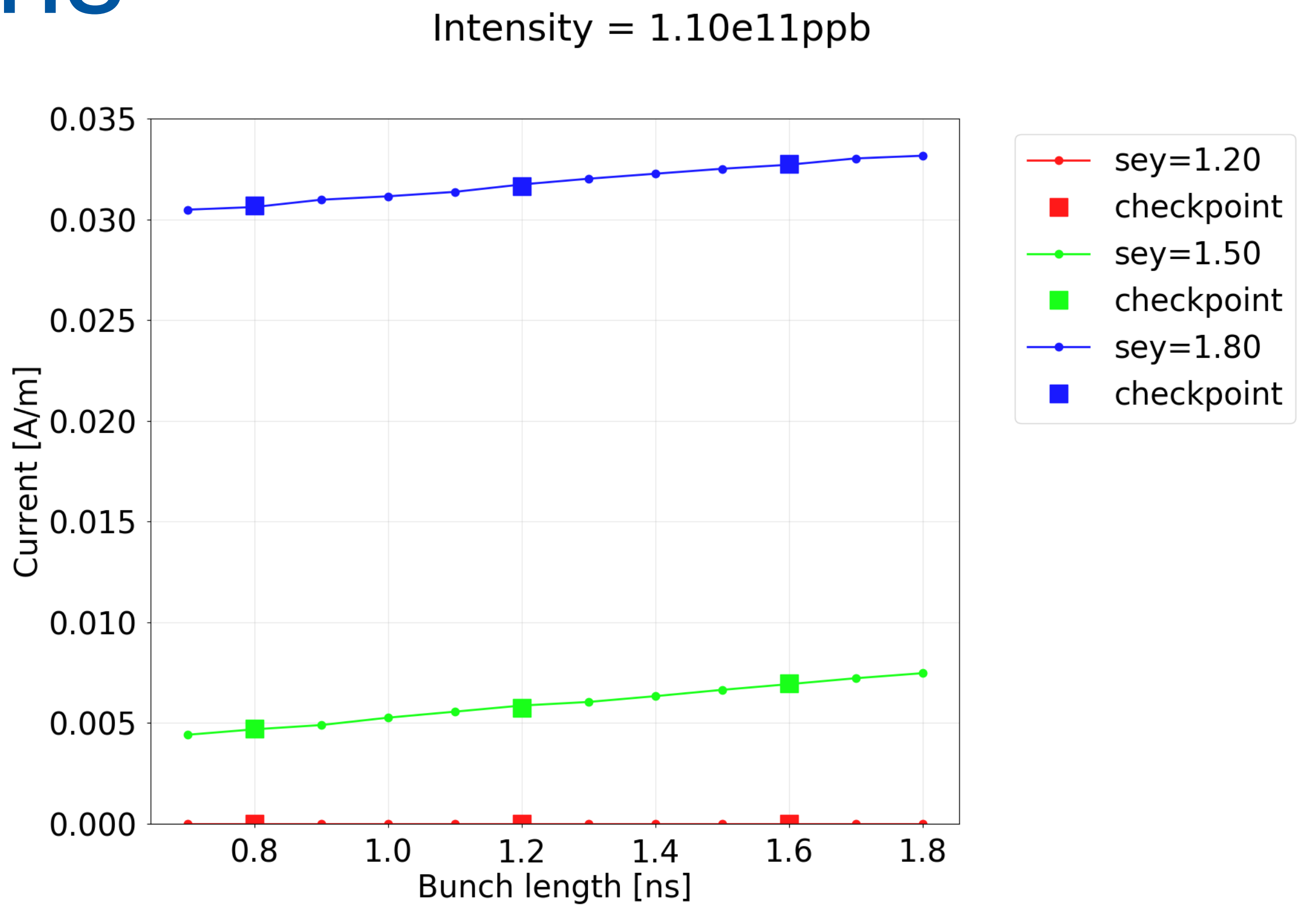
Comparison with previous simulations



- Simulation was killed here
- Upon restart it continues as if nothing happened

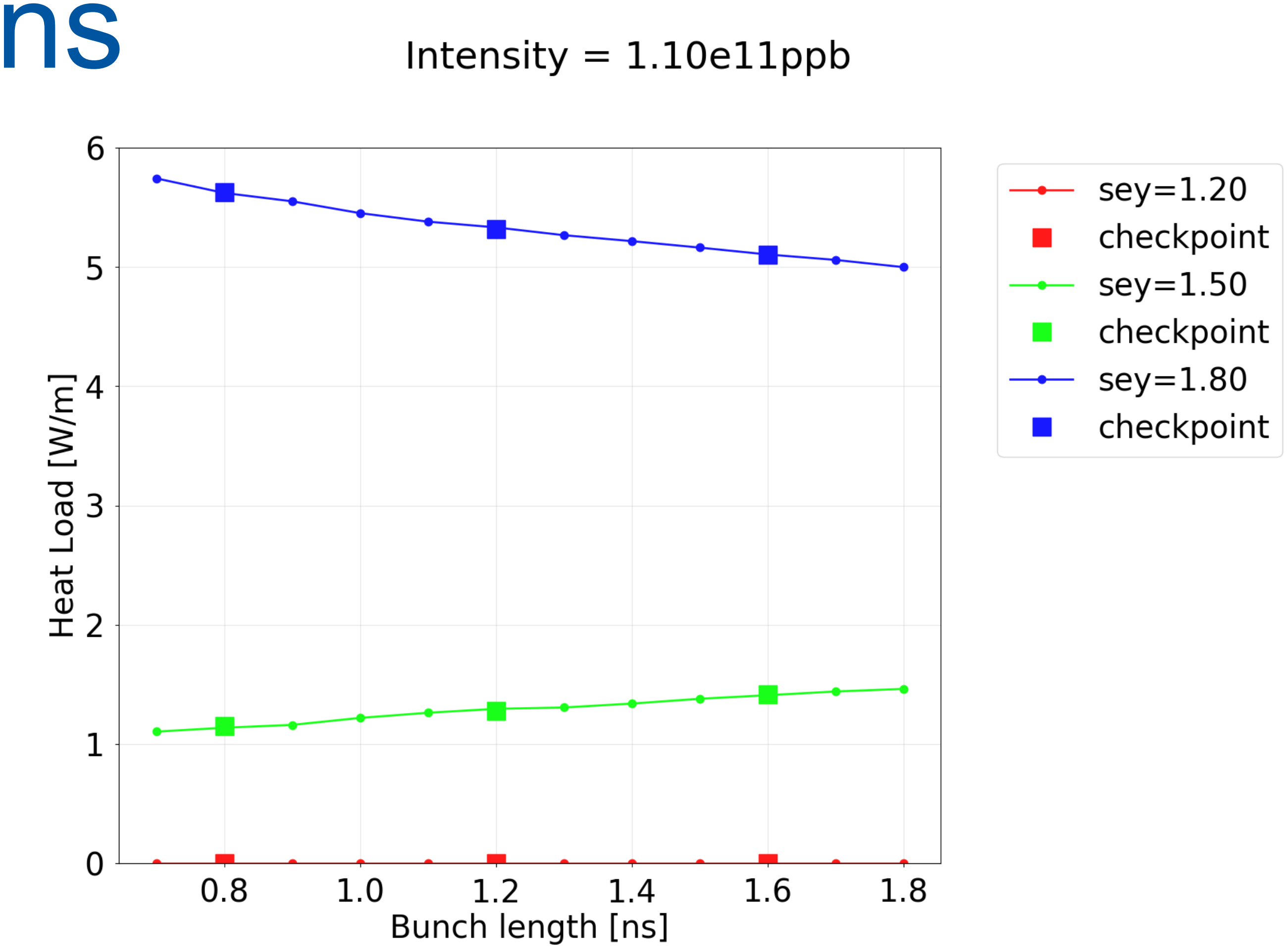
Comparison with previous simulations

- New checkpointed simulations agree with previous simulations

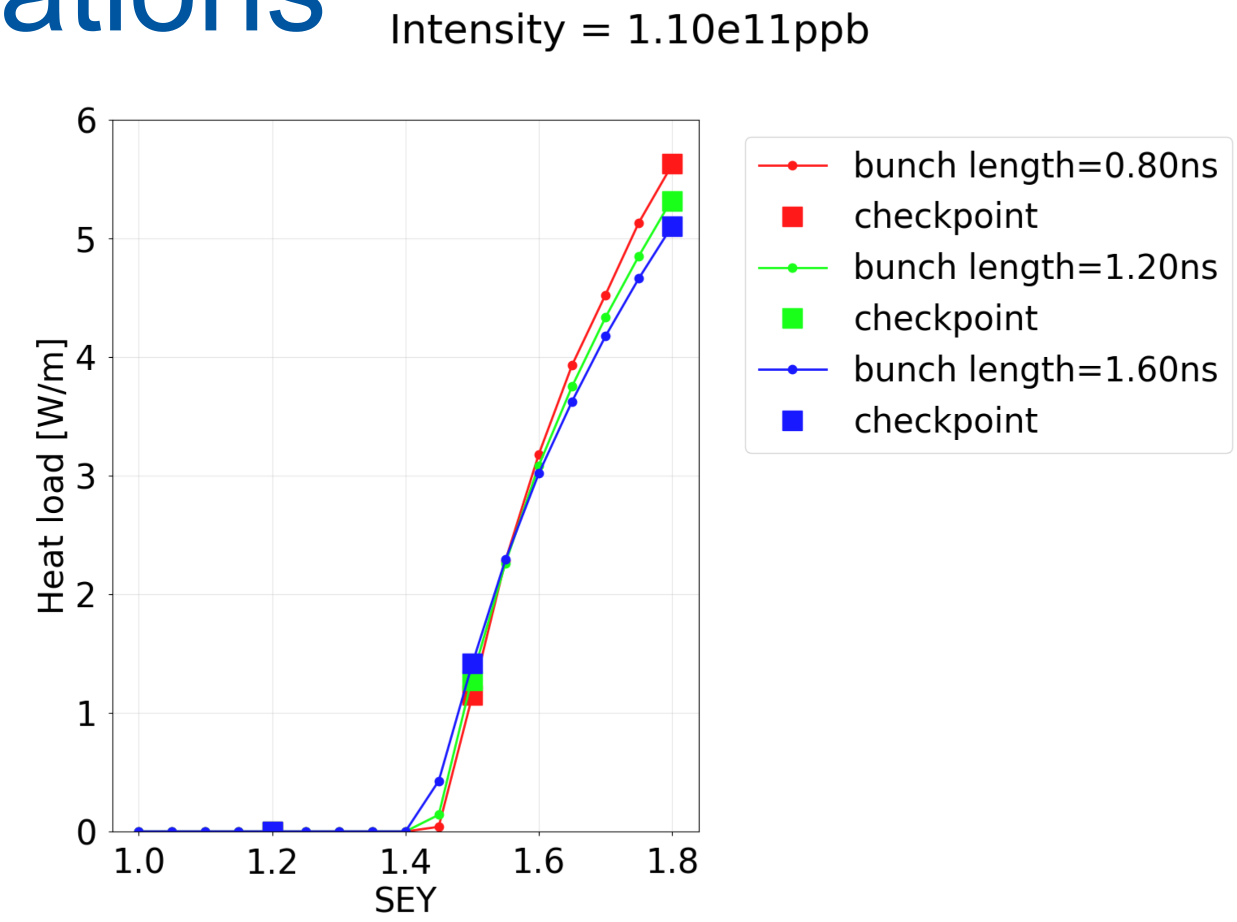


Comparison with previous simulations

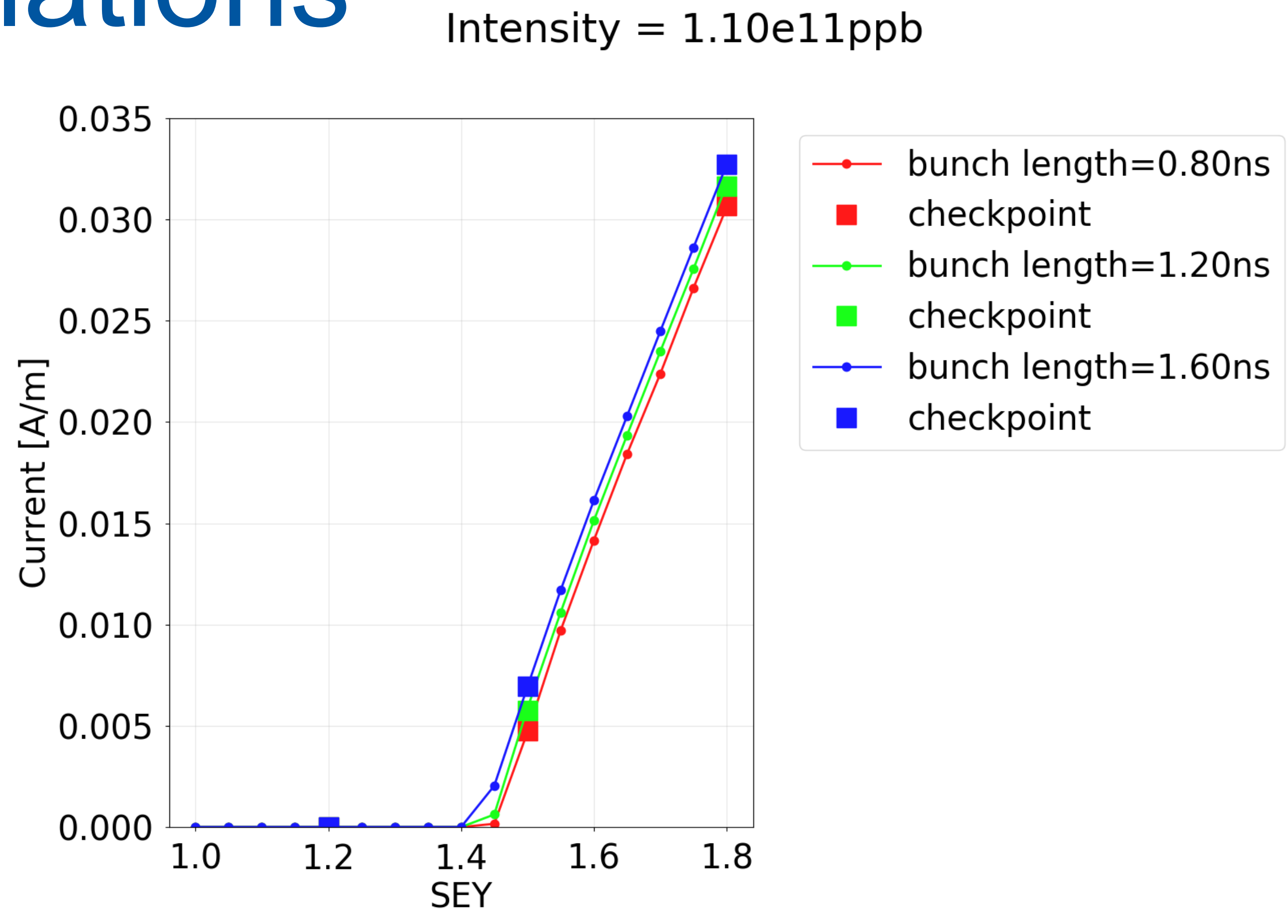
- New checkpointed simulations agree with previous simulations



Comparison with previous simulations



Comparison with previous simulations



Outline

- Goal
- How does it work?
- Comparison with previous simulations
- Summary

Summary

- A checkpointing feature was implemented in the PyECLOUD code
- The feature makes it easy to restart crashed simulations from the latest saved checkpoint
- Four new simulation input parameters
- It is included in version 7.6.0