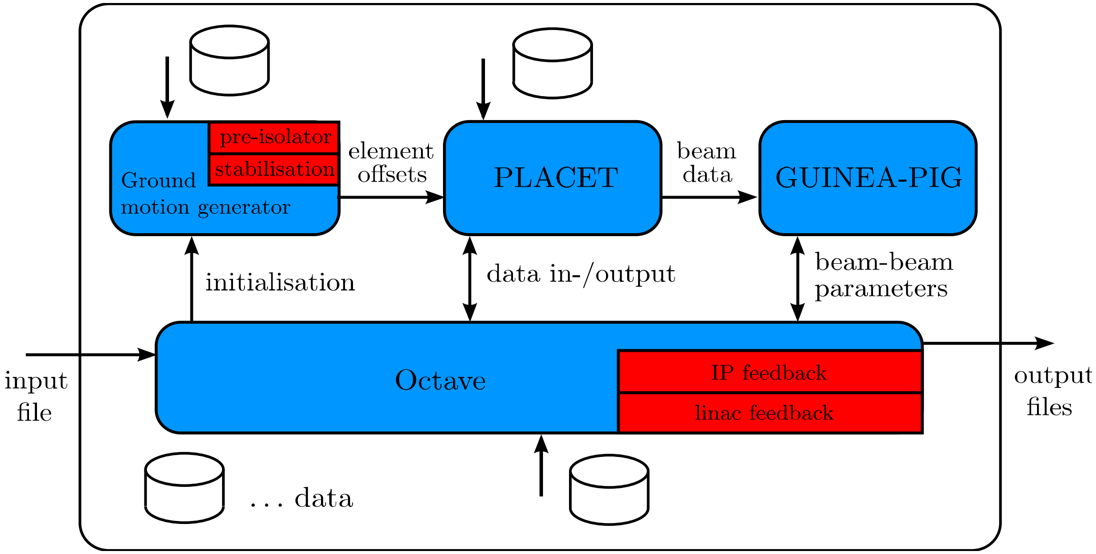
# LinSim

## Linear Accelerator Simulation Framework with PLACET and GUINEA-PIG

Jochem Snuverink (JAI, Royal Holloway)
Jürgen Pfingstner (CERN)

8th of October 2014

Content




1. Introduction
2. LinSim
3. Examples
4. Conclusions

# 1. Introduction

# Motivation

- Very good tools (PLACET, GUINEA-PIG) and lattice repositories are used in the section for simulations on ILC/CLIC and other linacs.
- However, several things are not automatized and are repeated by everybody:
  - Lattice and beam setup
  - Implementing simulations, which all have very similar structure
  - Debugging of code
  - Parallelization of simulations on a cpu-farm
  - Data storage and analysis
  - Backup and version control of complex simulations
- This causes that
  - Several versions of very similar code exist
  - Many versions are not up to date anymore after interface changes
  - Similar tasks are repeated by many people
  - Newcomers and external people have a hard time to get started

# Advantages of a general simulation framework

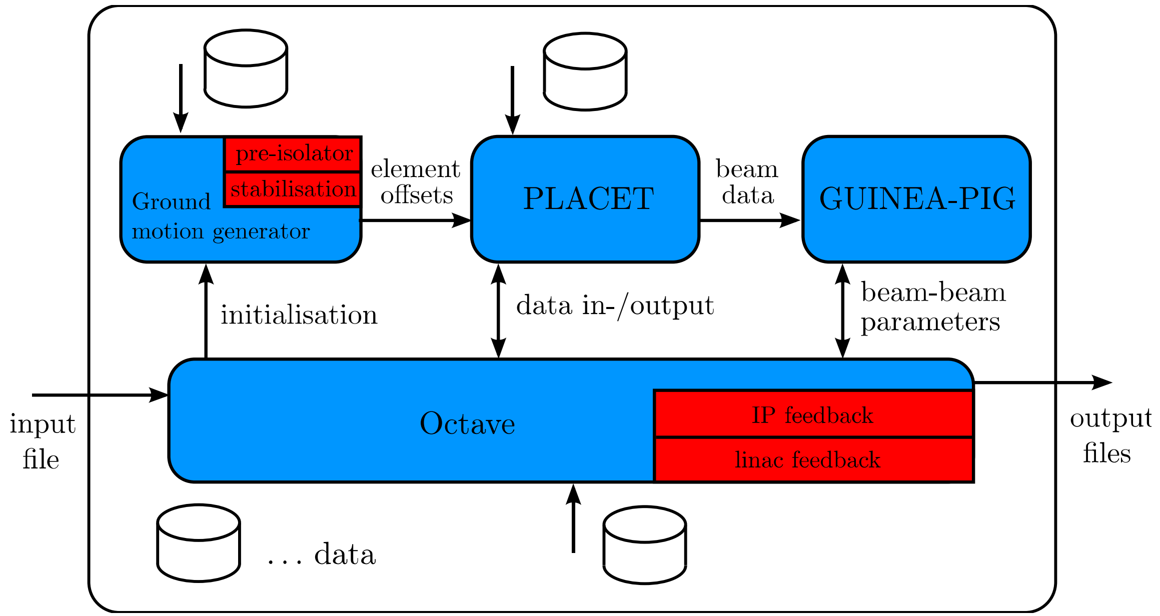
- Layer on top of the simulation tools that eases complex simulations.
- Lattice and beam setup is automatized.
- Most code only has to be written once (e.g. imperfections).
- Well debugged code.
- Features can be implemented that are too much effort for a single simulation (e.g. backup including versioning system)
- Data storage, parallel computing and analysis can be largely unified by providing tools.
- Increase in productivity

# 2. LinSim

# Features

- Lattice and beam setup for: CLIC, ILC, FACET, ATF2.
- Flexible simulation structure to implement most simulation scenarios efficiently.
- Many imperfections, feedbacks, steering algorithms are provided.
- Scripts for
  - Parallel computing on lxbatch (CERN batch computing service)
  - Data analysis tasks in Python and Octave (Matlab-compatible language)
- Documentation (work in progress - nearly complete).
- Consistency checks for settings and settings saving to reproduce results.
- Version control via svn
- Automated nightly testing (to be implemented).

# Internal structure



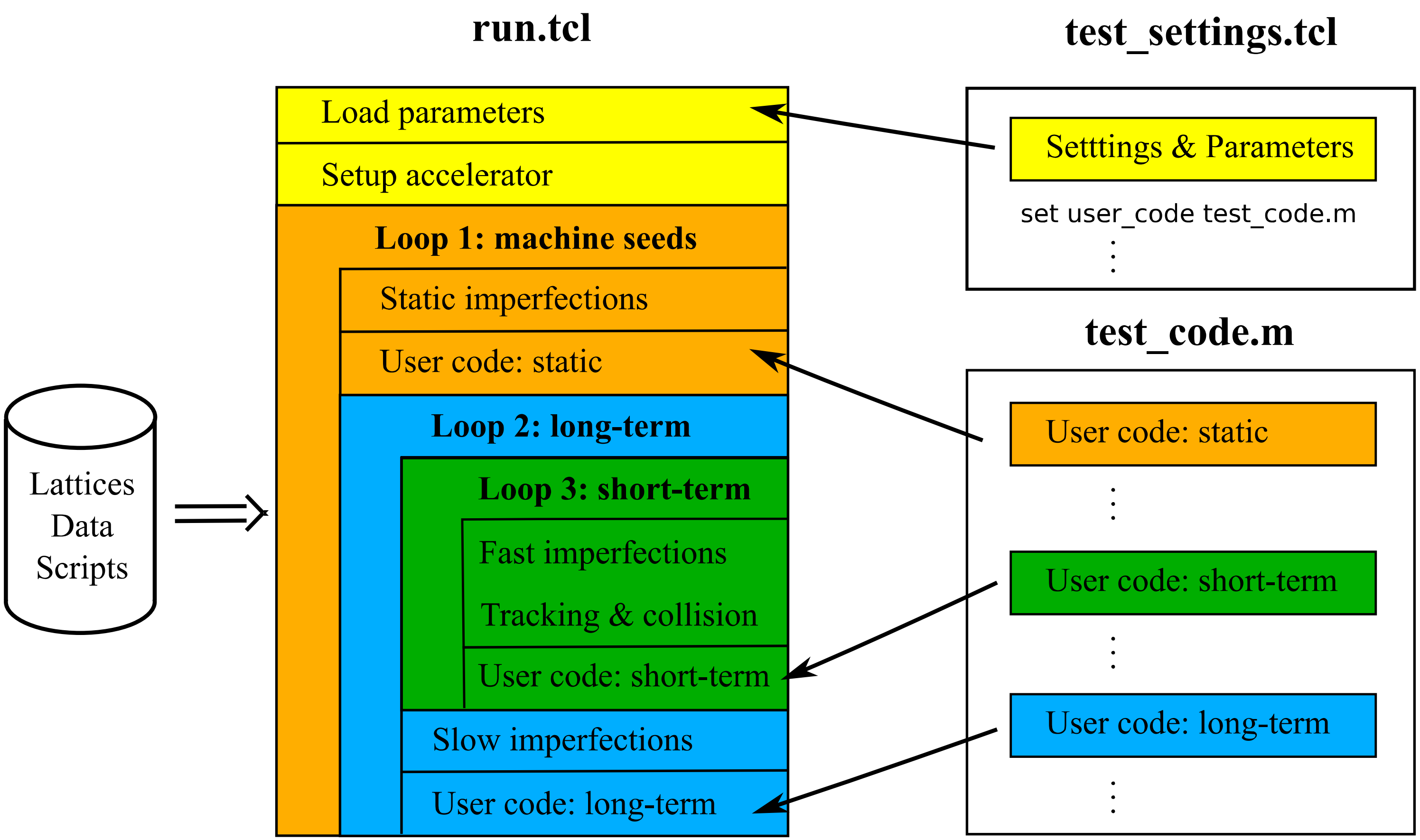

- LinSim interfaces PLACET and GUINEA-PIG via Tcl (mainly setup) and Octave (rest) scripts.
- Also external data are used: lattice files, ground motion models, reference orbit, …
- Input files control the behavior of LinSim fully.

# Simulation structure 1/2

- Simulations are structures into four parts:
  1. Initial setup: Lattice and beam creation, settings loading
  2. Short-term loop: pulse-to-pulse simulations
  3. Long-term loop: simulations on longer time scales
  4. Seed loop: For statistical averaging many “machines”

- Simulation consists of two parts
  1. LinSim base code
  2. Two test files that control and extend the simulation:
     - Settings file (in Tcl)
     - User code file (in Octave)

# Simulation structure 2/2



# Variable definition

- Universal variable names are used in LinSim
  - To be able to port code between accelerators
  - Quickly writing simulations
- Example:
  - BPM readings (bpm_readings)
  - Element indices (index_qd0)
  - Beamline names (“electron”, “positron”)
  - …
- Complete list in the manual

# 3. Examples

# Example 1: QD0 roll scan (CLIC) 1/2

```
set user_code "tests/qd0_rollscan_code.m"
set machine_name "CLIC"
set dir_name "QD0_rollscan"

set values_to_scan {-100 -75 -50 -25 0 25 50 75 100 0}
set nr_time_steps_short [llength $values_to_scan]
set nr_time_steps_long 1

set use_main_linac 0
set use_bds 1
set use_beam_beam 1
```

**qd0_rollscan_settings.tcl**

---

```
if (sim_mode == static_mode)
    % store initial roll
    qd0_roll_start = placet_element_get_attribute('electron', index_qd0(electron), 'roll')
end

if (sim_mode == short_term_mode)
    % new roll setting
    qd0_roll = values_to_scan(time_step_index_short) + qd0_roll_start
    % apply to beamline
    placet_element_set_attribute('electron', index_qd0(electron), 'roll', qd0_roll)
end

if (sim_mode == long_term_mode)
    % nothing to be done here
end
```

**qd0_rollscan_code.m**

# Example 1: QD0 offset scan (CLIC) 2/2

**Start simulations in PLACET:**

placet run.tcl tests/qd0_rollscan_settings.tcl

**Data analysis in Python:**

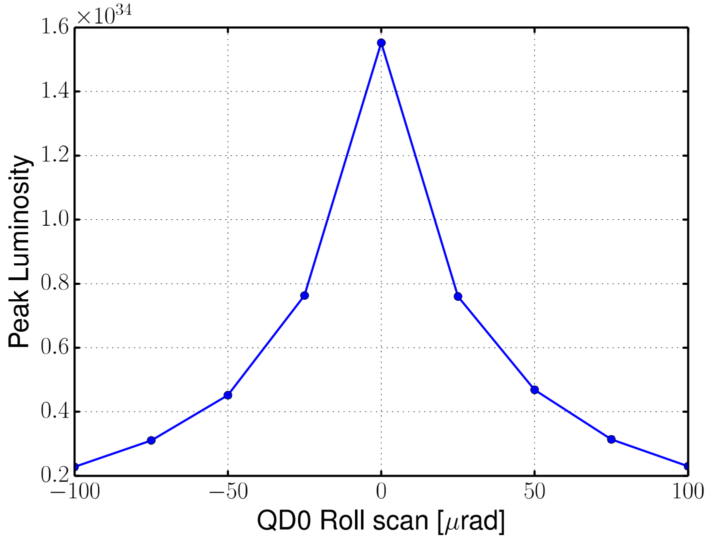
python

import TrackingAnalysis

a=TrackingAnalysis.MeasurementStation(directory="../QD0_rollscan/”)

a.lumiScanPlot(-100,100,25,label='QD0 Roll scan [$\mu$rad]',plotname='QD0Roll’)

Result of analysis:



# Example 2: Response matrix calc. with ground motion influence (FACET) 1/3

```
set user_code "tests/facet_response_matrix_demo_code.m"
set machine_name "FACET"
set dir_name "ResponseMatrix"

# first argument is seed number
set global_seed [lindex $argv 1]
set nr_time_steps_short 23
set nr_time_steps_long 2

# static misalignment
set use_misalignment 1
set alignment_bpm_sigma 10 ;# [um]
set alignment_dipole_sigma 10 ;# [um]

# ground motion generator short term
set ground_motion_x 1
set ground_motion_y 1
set groundmotion(model) "B"

# ATL motion
set ground_motion_long_x 1
set ground_motion_long_y 1
set delta_T_long 3600

# Response parameters
set corr_step 1 ;# [m]
```

- Studies the influence of ATL motion on the measurement of the response matrix after one hour.
- During the measurement ground motion of model B is used.
- Initial normal distributed misalignments are applied.

**facet_response_matrix_settings.tcl**

# Example 2: Response matrix calc. with ground motion influence (FACET) 2/3

```
if (sim_mode == static_mode)
    % apply misalignment --> already in Framework!

    R_x = zeros(nr_bpm, nr_corr);
end

if (sim_mode == short_term_mode)
    % set corrector
    placet_element_set_attribute('electron',corr_index(time_step_index_short), 'strength_x', corr_step);
    if (time_step_index_short > 1)
        placet_element_set_attribute('electron',corr_index(time_step_index_short-1), 'strength_x', 0);
    end

    if (time_step_index_short == 1)
        % ref orbit
        ref_orbit_x = bpm_readings(:,x,electron);
    else
        % store results
        R_x(:,time_step_index_short-1) = (bpm_readings(:,x,electron) - ref_orbit_x)/corr_step;
    end
end

if (sim_mode == long_term_mode)
    % apply ATL misalignment --> already in Framework!

    % save response matrix
    filename = ['R_x', int2str(time_step_index_long), '.dat' ]
    save(filename, 'R_x', '-ascii');
    % reset for next iteration
    R_x = zeros(nr_bpm, nr_corr);
end
```

**facet_response_matrix_code.m**

# Example 2: Response matrix calc. with ground motion influence (FACET) 3/3

**Start simulations in PLACET:**

placet run.tcl tests/facet_response_matrix_settings.tcl 3

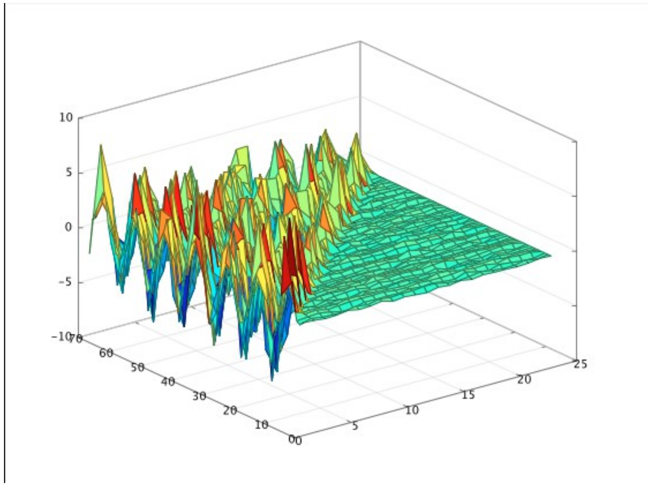
**Example of seed scan on lxbatch (on AFS):**

cd jobs

./submit_jobs_seed_scan.sh

tests/facet_response_matrix_settings.tcl 8nh 1 20

First response matrix:



# How to get LinSim?

- PLACET and GUINEA-PIG (if beam-beam is calculated) need to be installed.
- LinSim is included in a larger svn repository. It is easiest to check out the full CLIC directory with

  *svn co svn+ssh://[username]@svn.cern.ch/reps/clicsim/trunk/LinSim*

- If also other accelerators are simulated also the according directories have to be checked out. But not all files are necessary (see the documentation).
- Documentation is located in *LinSim/doc/Framework_doc.pdf.*

# 4. Conclusions

- A framework for linear accelerator simulations was presented
- A layer on top of PLACET and GUINEA-PIG
- It provides scripts and algorithms for complex simulation tasks
- Especially useful for newcomers

Not something complete! Everybody is welcome to suggest and implement. For any help/questions, please contact us!

Thank you for your attention!