Flight Simulator for ATF2

Glen White (LAL / SLAC)

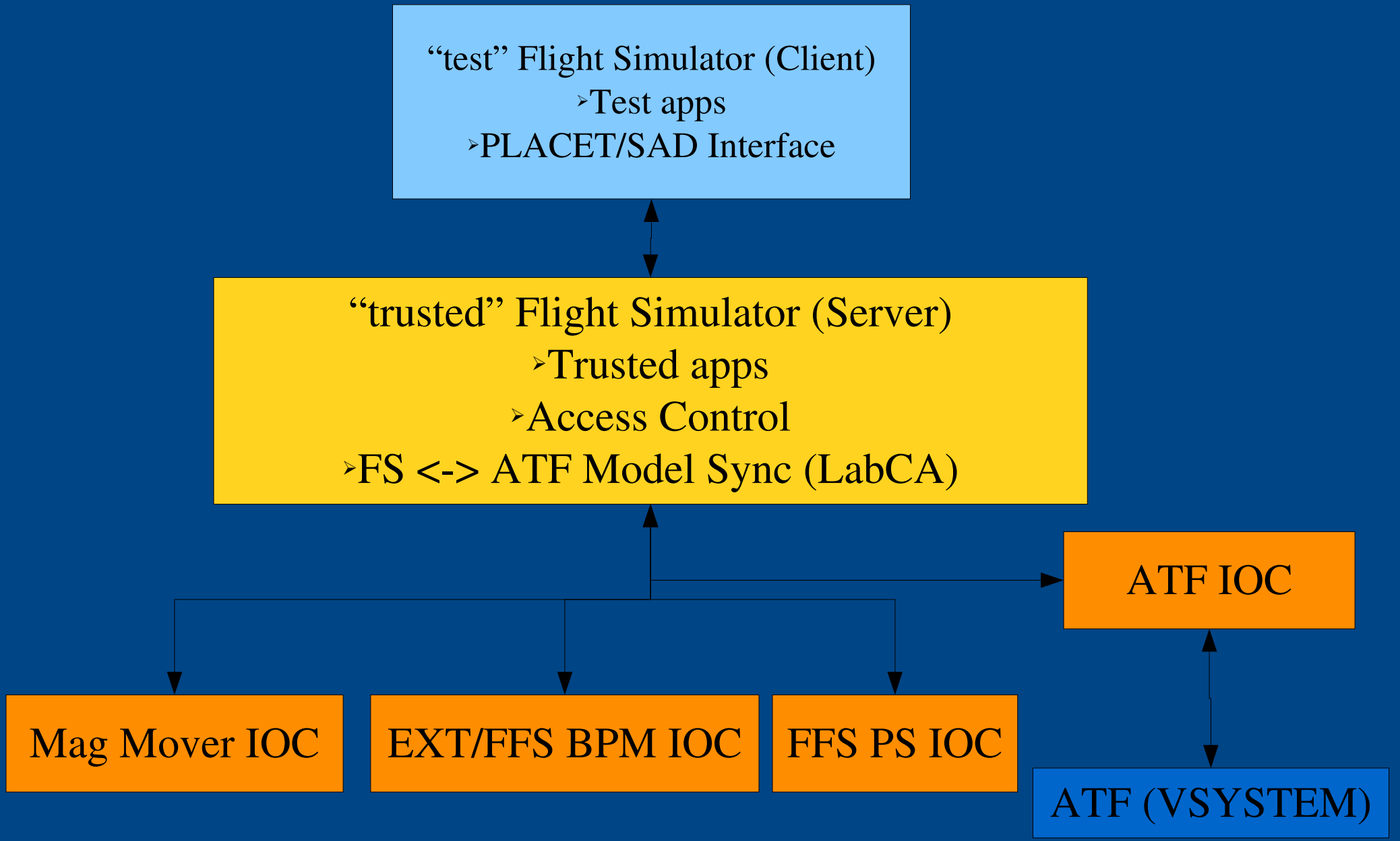
ATF2 Software Workshop, LAL, June 2008

- Project goals.
- Operation overview.
- Tests at ATF.
- Software details.

Flight Simulator Goals

- Provide simple to use, beam dynamics oriented, portable control access framework for ATF2 tuning tasks.
- Simple and reversible transition from beam dynamics simulation to accelerator-ready code.
- Ability for international collaborators to develop beam tuning tools without need for expert-level knowledge of control systems.
- Flight simulator operates in simulation mode at external location in the same way as the production system deployed at ATF2.

Implementation at ATF(2)



Software Overview

- Core Flight Simulator software written in Matlab, based on Lucretia accelerator toolbox.
- Floodland module provides flight simulator functionality:
  - Server-client communications.
  - Access control to client installations.
  - Server-based data services (bpm averaging etc).
  - Sync epics ioc db entries with Lucretia model.
  - Maintain updated AML and Lucretia models.
  - Provide PS setting, magnet move functionality through native Lucretia functions (PSTrim, MoverTrim).

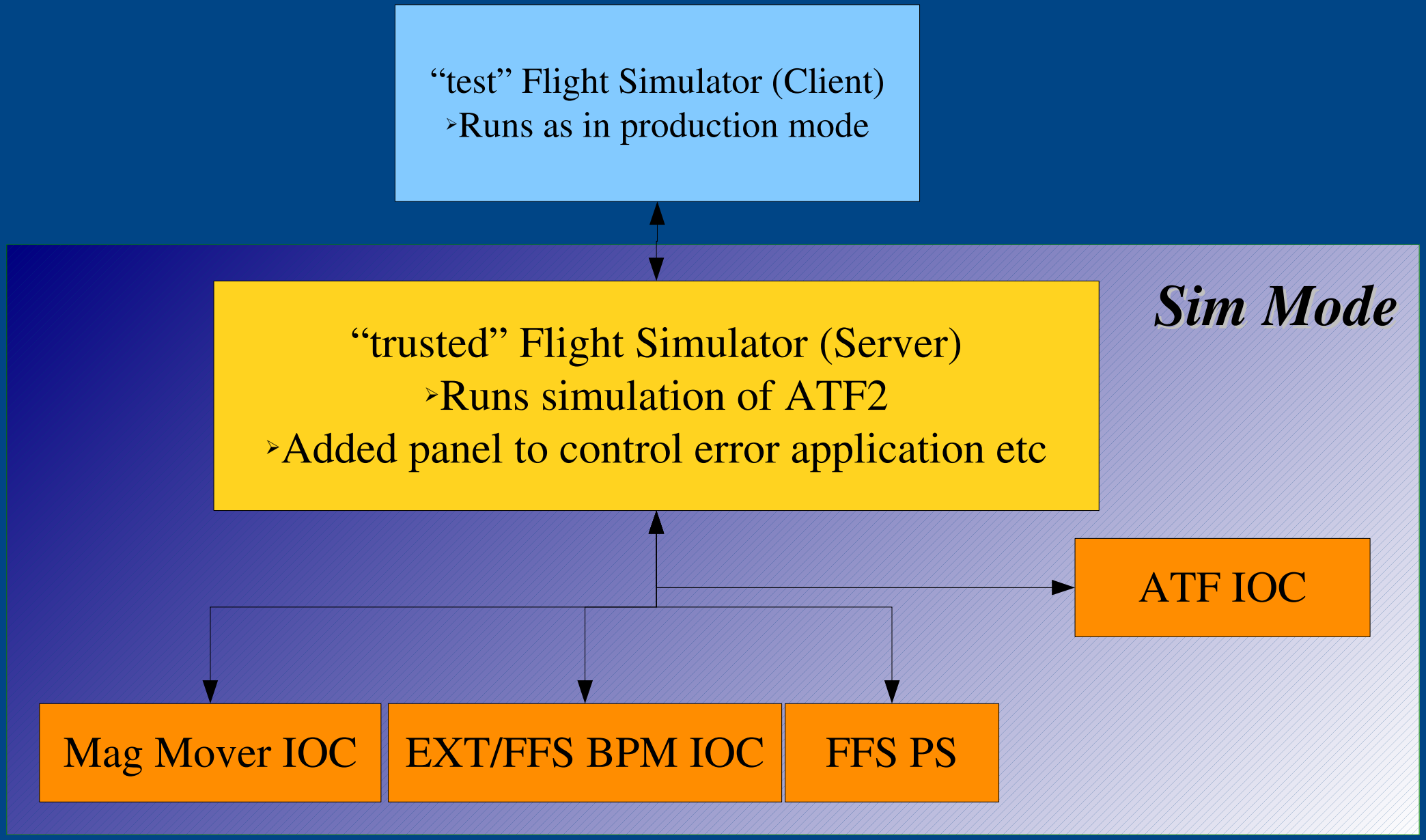
Software Overview

- Lucretia2AML
  - Synchronises Lucretia model with AML
  - Entry-point for non-Lucretia tracking code support
- Client-side runs server for non-Lucretia based support through socket communications
  - Currently tested with PLACET (running with tcl/tk & octave)
- EPICS (hardware IOC's + simulation)
    - Movers, bpms, power supplies
    - ATF interface (Vsystem –ring mags + bpms, ATF controls)
- Documentation effort starting
  - *https://confluence.slac.stanford.edu/display/ATF/ATF2*

Security

- The computer which hosts the trusted server on control system sub-net has read/write privs on all EPICS databases.
- Users test their code on client which must request access privs from trusted server.
  - When tested, code can be migrated easily from client to server which then sits on main control console.
- Security provided by ATF network infrastructure and by EPICS access control which only allows write access through trusted server.

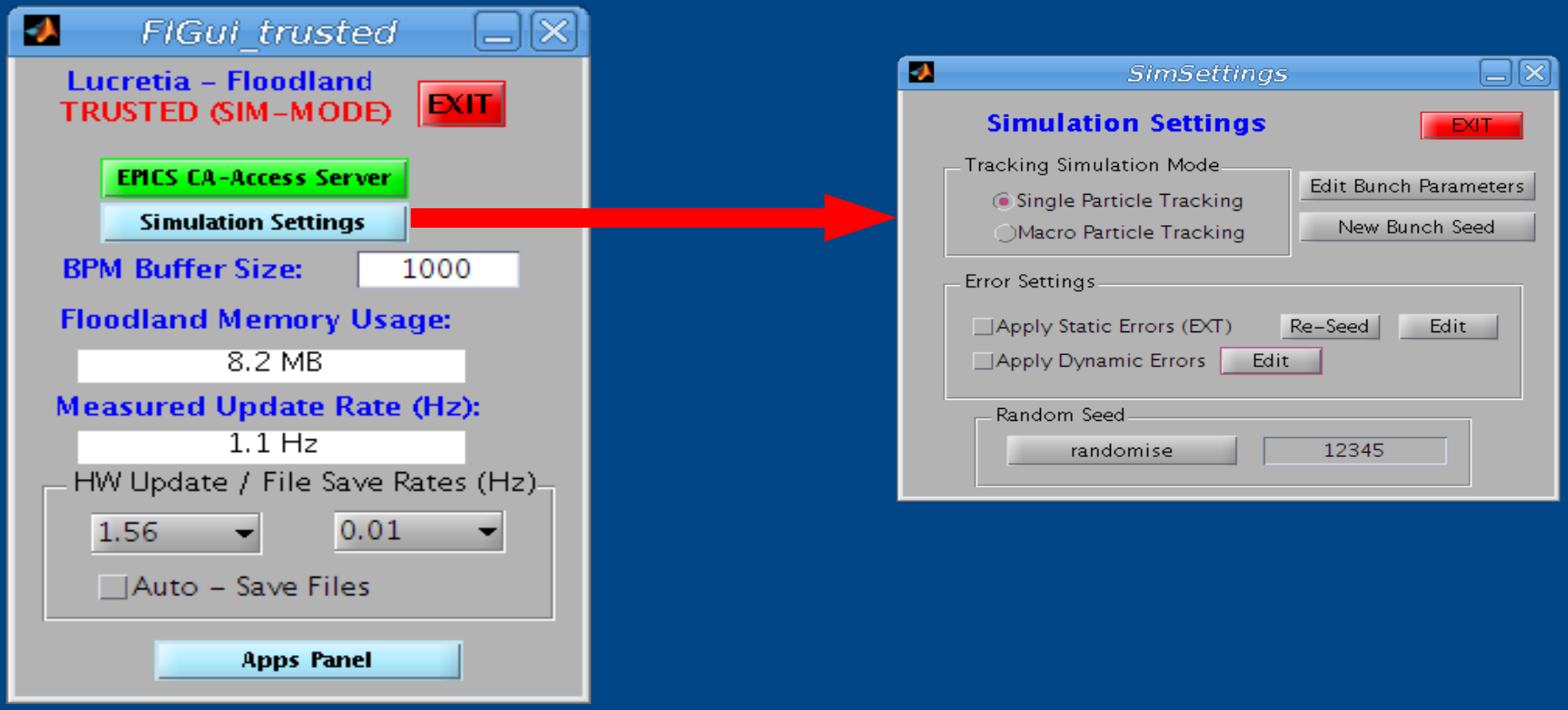
Sim. Implementation



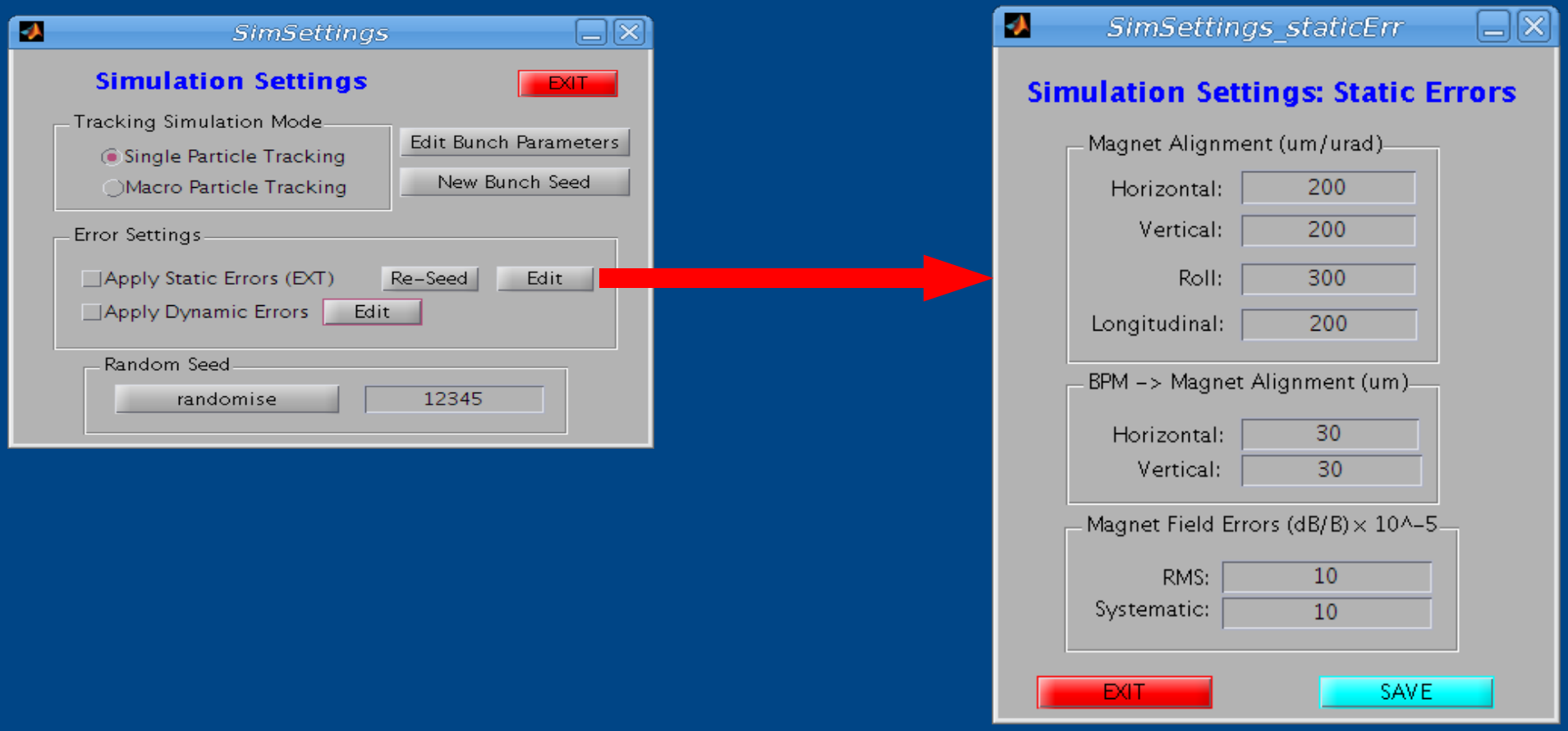
Simulated Functionality

- Make as similar to production system at ATF as possible.
- Runs all on single pc (or more if desired).
- Enables simulation of real machine, enabling realistic testing of accelerator-ready beam dynamics code without need of access to KEK systems.
- EPICS IOCs are also running in the simulated environment providing the ability to build and test custom hardware interface with a simulated version of the complete set of in-production IOC's, control software and tuning routines etc.

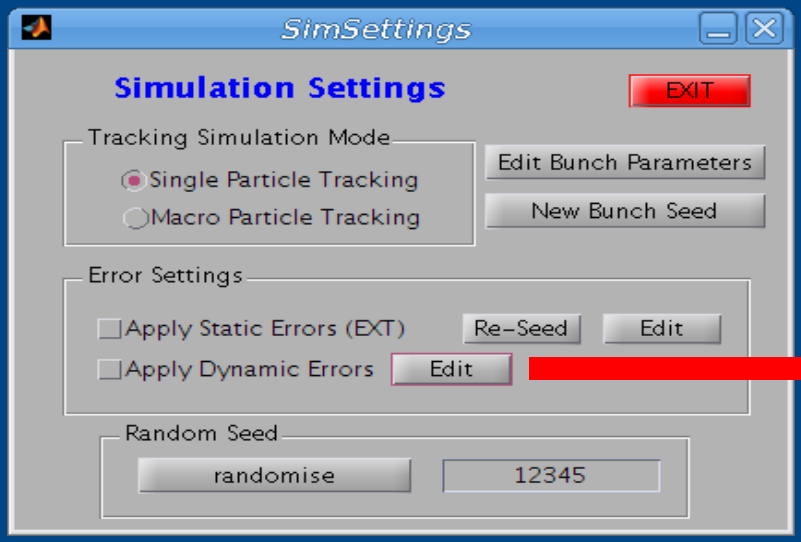
Simulation Mode

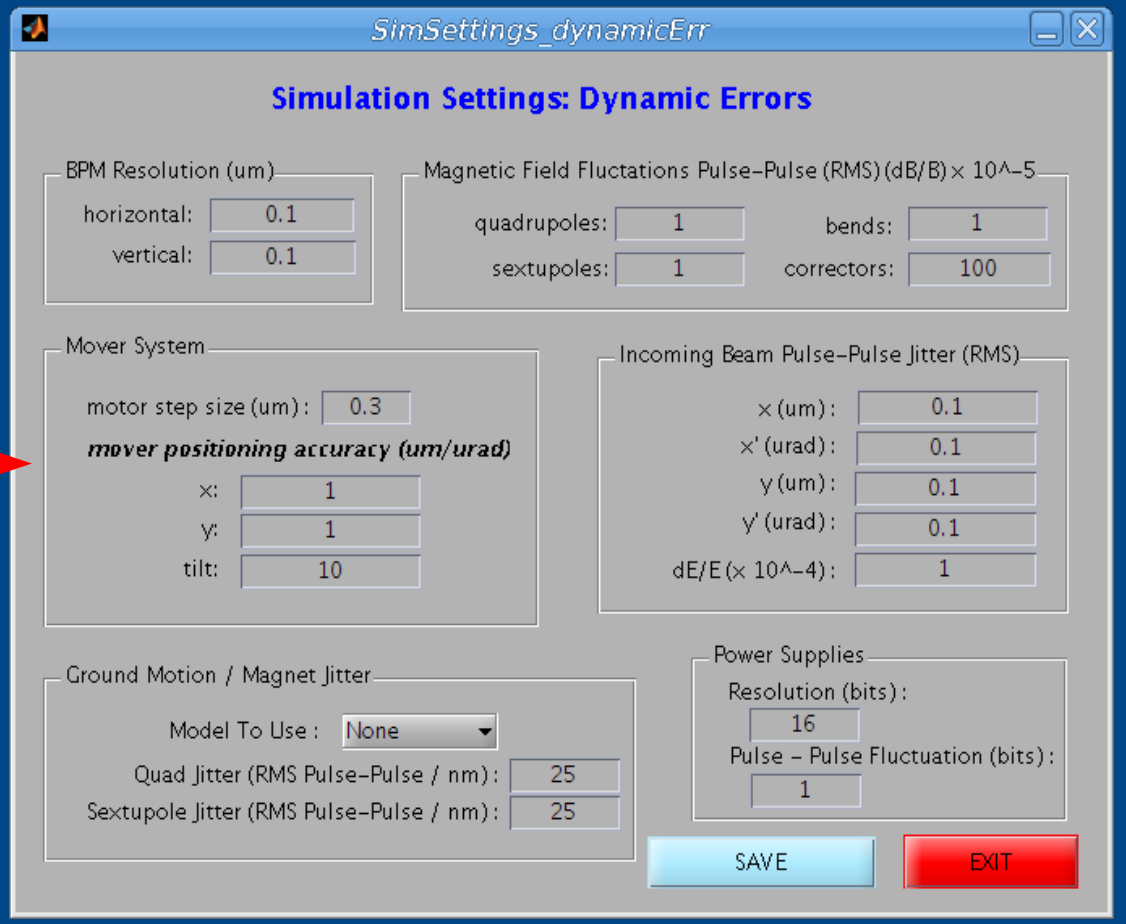


Simulation Mode



Simulation Mode





External Interface

- Client Flight Simulator installation runs a socket server to provide Lucretia and Floodland commands to external beam dynamics software
  - PLACET, SAD etc...
- Only requirement is that the external software package implements read/write access to a tcp socket.
- Can read in current live lattice at any time using AML file. (then update via this or through the FS client server).
- Example generic c++ implementation included in flight simulator software package *(Lucretia/src/Floodland/testClient/test_connect.cpp)*.
- Read/write interface done using AML-like commands.

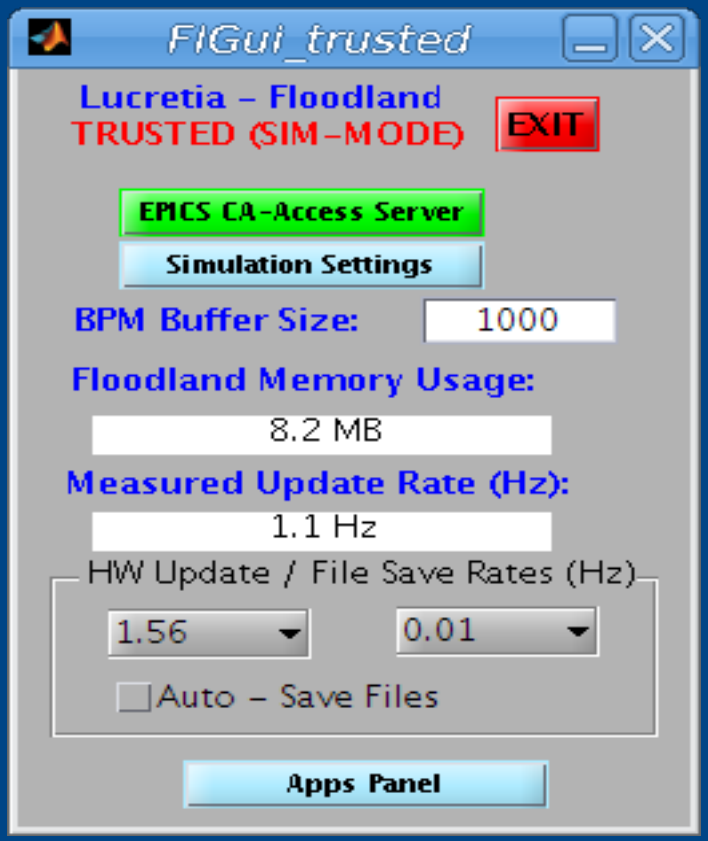
Example Server Commands (external interface)

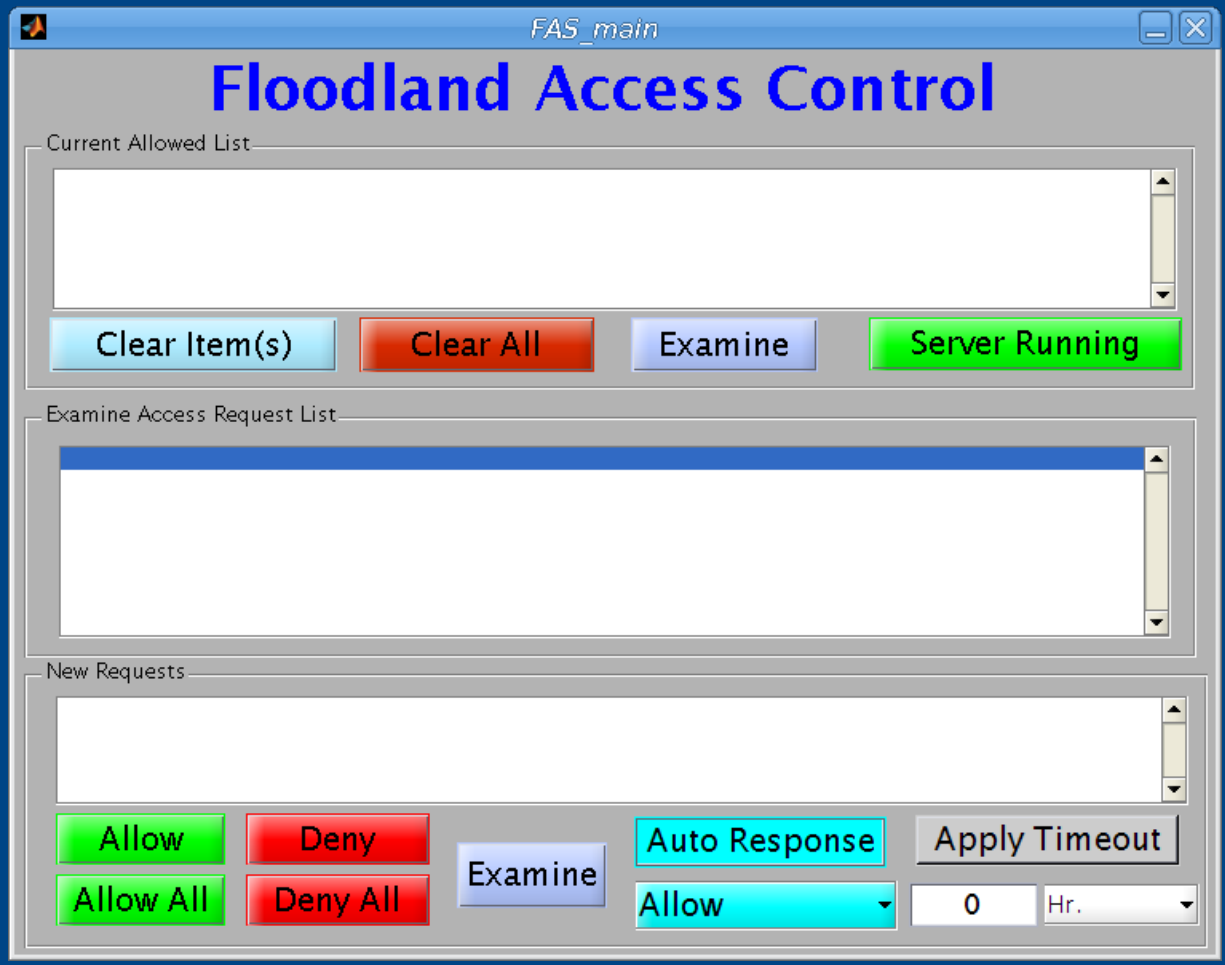
- Get/set power supply settings for magnets
  - *fl_socket << "amlget('PS1:design:num','PS2:design:num','PS5:design:num');";*
  - *fl_socket << "amlset('PS1:design:num',1.2,'PS2:design:num', 0.96,'PS5:design:num',0.5e-3);";*
- Get average orbit from last 20 bpm pulses, applying default quality cuts
  - *fl_socket << "bpmave(20,1);";*
- Request control access
  - *fl_socket << "access-request PS39 PS40;”;*

Test Implementation at ATF during May run

- Same basic software setup as for ATF2.
- Only ATF IOC used (access to ATF magnet power supplies and bpms).
- Write and run some test applications
  - EXT steering (PLACET)
  - EXT bumps (PLACET)
  - Online EXT dispersion measurement.

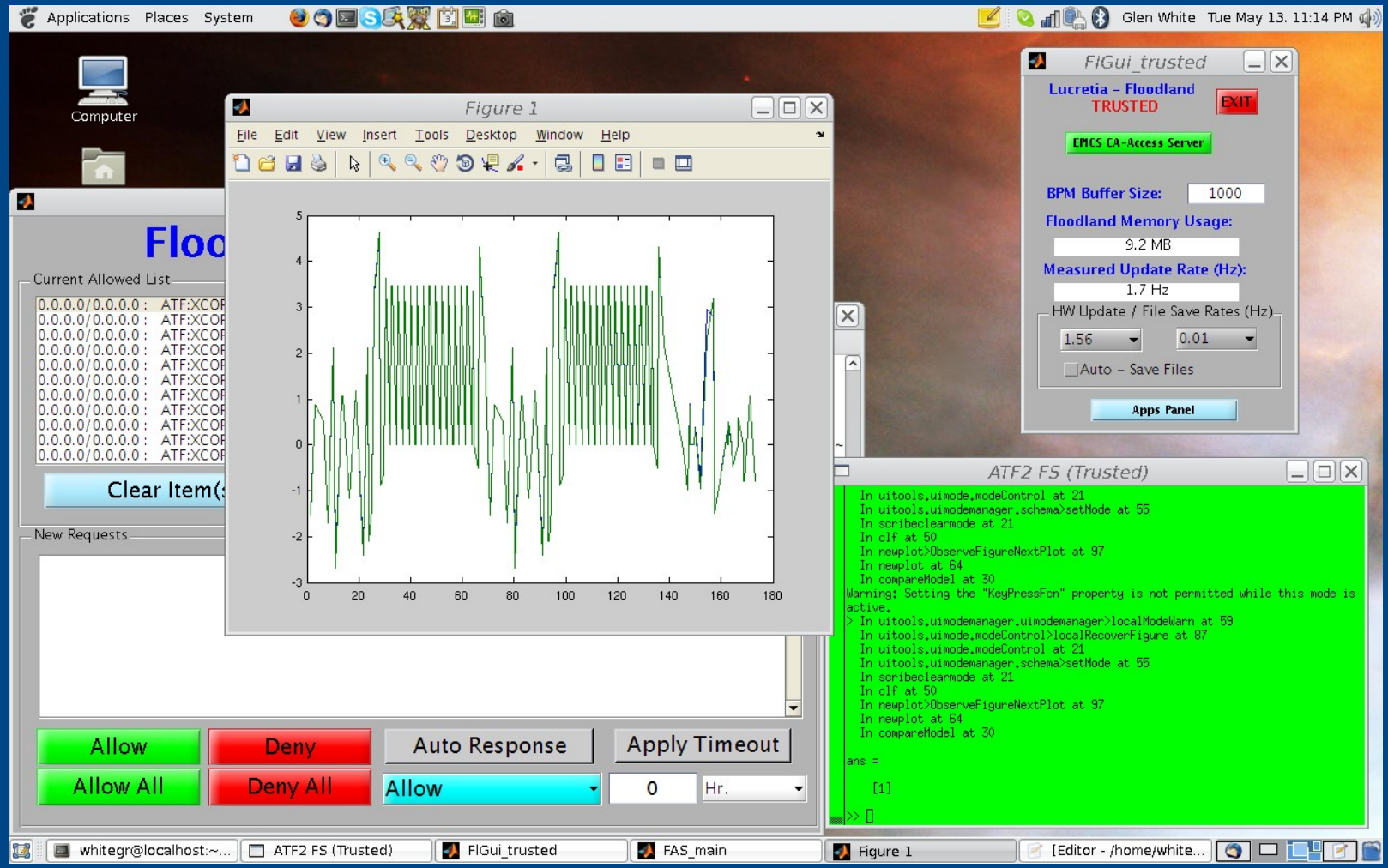
“trusted” FS (server) runs on *atfsad.atf-local*





“test” FS (client) runs on user laptop (@atf-local)

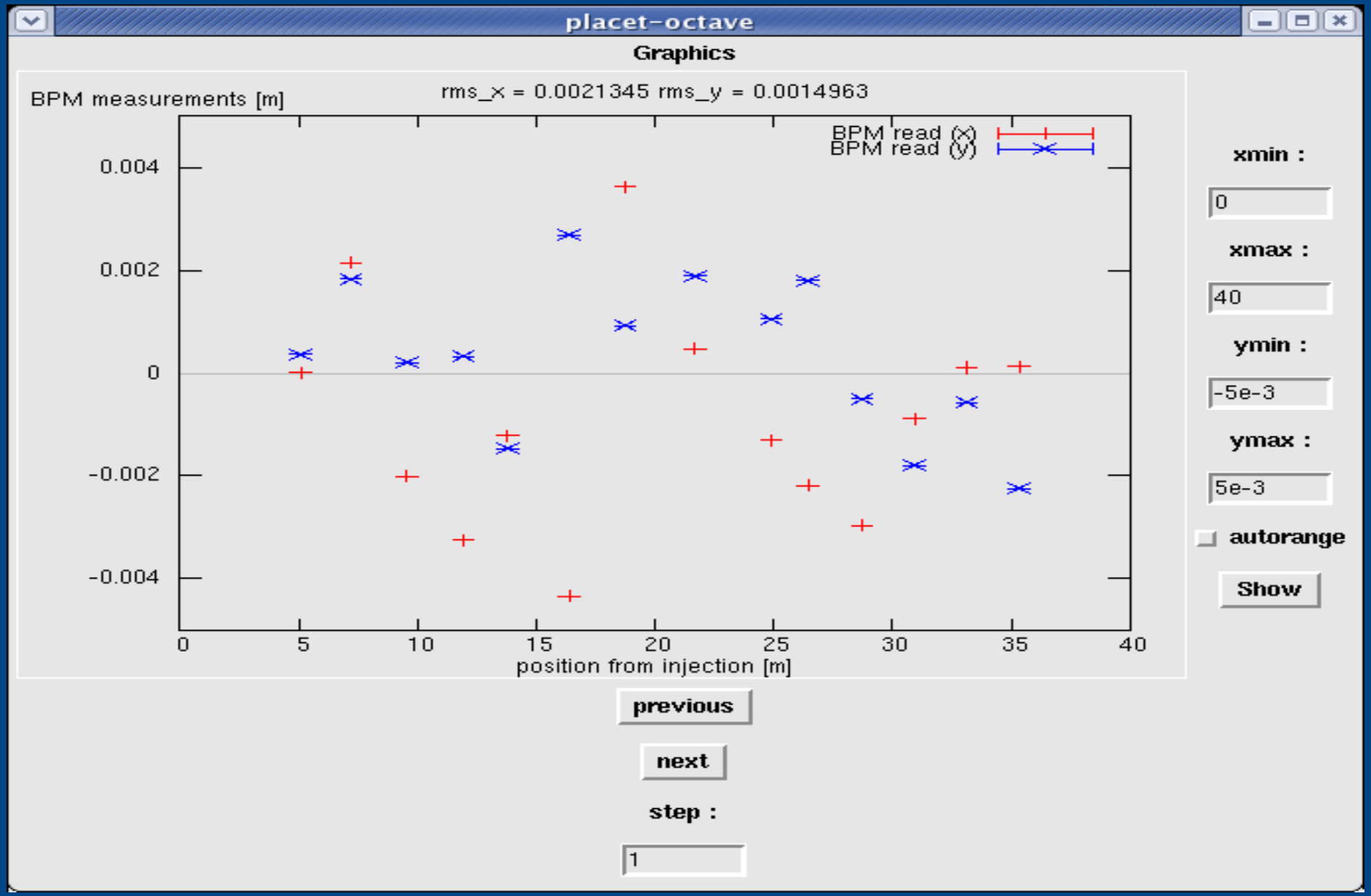
Live vs. Simulation Model Comparison



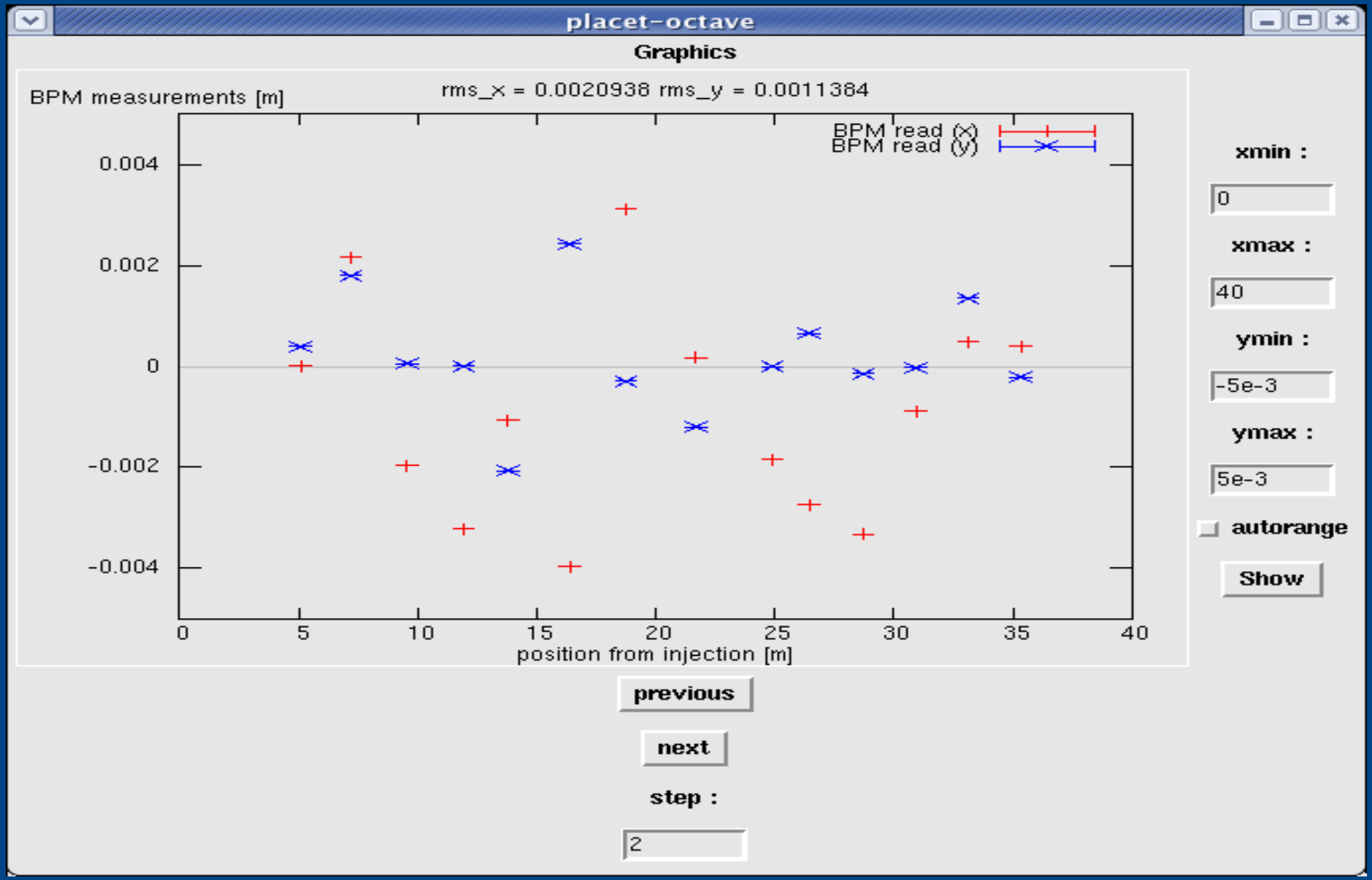
PLACET Steering/Bump Tests

- Using PLACET interface on “test” flight simulator software installation.
- Measure R12 and R34 elements between correctors and BPMs in EXT.
- Use Yves' “1-all” correction scheme to try to improve EXT orbit in x and y.
- Use bump calculator to insert orbit bump in 2 bpms for test.

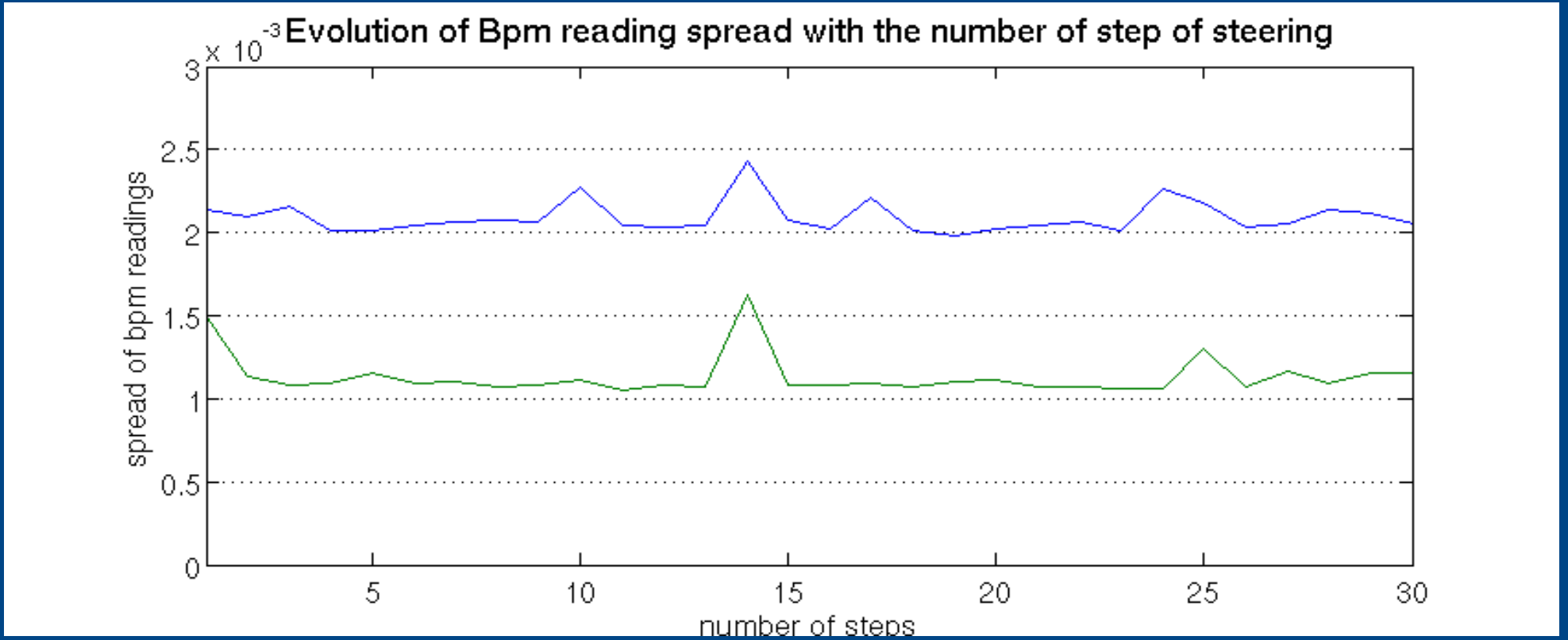
Orbit Before Correction



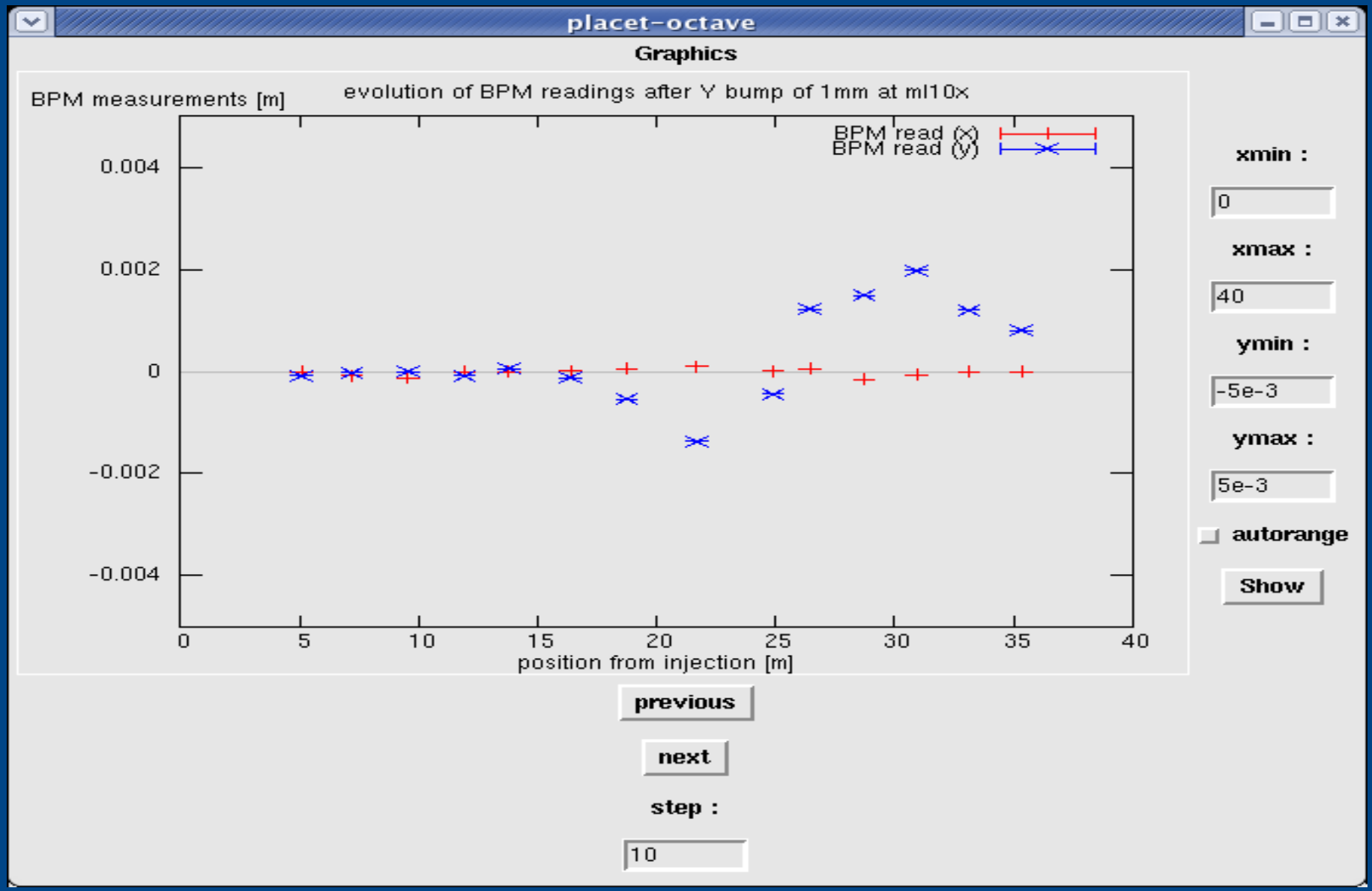
Orbit After 1 Correction



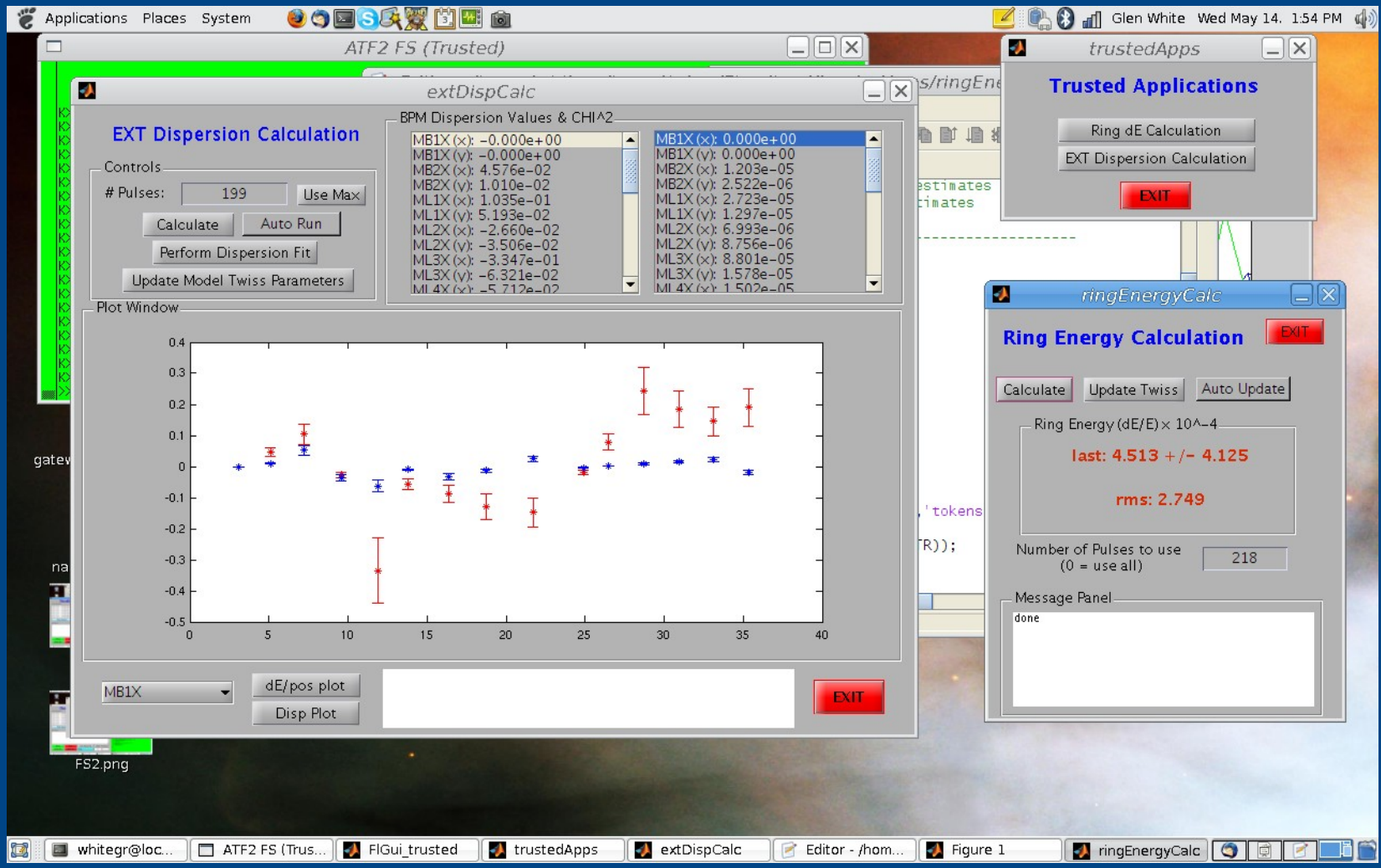
RMS Orbit During Corrections



1mm Bump @ ML10X



Disp Meas Tests



Software Details

- The following concentrates on the Matlab/Lucretia interface.
- Yves' talk will concentrate on an example third-party code interface (PLACET).

Floodland Data Structures

- Standard Lucretia globals
  - BEAMLINE (full lattice list)
  - GIRDER (Movers)
  - PS (Power supplies)
- Added by Floodland module for FS:
  - INSTR (bpms, wirescanners etc...)
  - FL (includes stuff like DataStore, SimModel, GUI handles)
- Can examine interactively using live Matlab command window, or using the graphical *BeamlineViewer* browser.
- Also available to other environments through Client Server.

Floodland Data Structures

- BEAMLINE cell array contains data that corresponds to Model.
- PS and GIRDER reflect changes to the default model (current running conditions).
  - i.e. To get true current magnet strength: *BEAMLINE{N}.B*PS(BEAMLINE{N}.PS).Ampl*
- Data sharing between Lucretia and non-Lucretia code:
  - FL.DataStore.PLACET.placetApp.someData
  - FL.DataStore.LUCRETIA.lucretiaApp.otherData
  - Non-Lucretia access through FS Client Server.

Some Floodland Core Functions

- AccessRequest
- compareModel
- FlAssignHW
- FlHwUpdate
- MoverTrim
- PSTrim
- useApp

See wiki Documentation or Matlab help system for details

Steps for Creating a User Application

- Create a sub-directory with the name of your application in Lucretia/src/Floodland/testApps/.
- Copy all your files into the sub-directory (if you use further sub-directories, be sure to properly handle path settings etc such that the main Floodland environment works).
- You should have a file in your directory with the same name as the directory which is the entry point to your app.
  - If the entry file is a .m file- it is assumed that this is a subroutine (of same name), and when the "run app" button is pushed, this subroutine is simply called (with no arguments).
  - If the entry file is a .fig file, it is assumed that this is a Matlab GUI and run appropriately.
- Your application will now appear in the pull-down menu on the client flight simulator GUI and can be run with the "run app" button

Migrating from simulation to flight simulator

- TrackThru commands- instead of getting BPM readings through Lucretia tracking, use the INSTR data array. NOTE: TrackThru can still be used to track a bunch through the live version of the Lucretia lattice for simulation comparison to real data etc.
- Issue FlHwUpdate; statements internally to your application every time you expect the state of the machine to have changed.
- Remember to leave appropriate periods of time in the real world for changes to happen, perform checks to see if the magnet you thought you moved actually moved etc.
- You need to apply a second argument to PSTrim and MoverTrim commands to indicate that you wish to use the versions which actually apply the changes to hardware (second argument should be 1).
- You need to request access to any device you want to change ahead of time with *AccessRequest* command.

Writing Graphical Interface for User App

- Design layout with Matlab *GUIDE* application (buttons, plot axes, edit boxes etc...).
- Edit callback functions in <appname>.m file to take the appropriate actions.
- Small changes to default m-files to get GUI to function in FS- detailed in wiki documentation.
- Use e.g. get(FL.Gui.<appname>.edit1,'String') to get the contents of the edit field named edit1 in another function.
- To run some other function periodically (to keep your GUI fields updated for instance):
  - Attach it to a Matlab timer object
  - Register it to be run whenever the FS updates
    - In <appname>_OpeningFcn :
    - FL.userfun.<appname>_myFun='myFun'
    - FL.userdata.<appname>_myFun={<list of variables to pass to myFun>}
- More details in wiki documentation

Hardware Development

- Having a complete, local copy of the FS control software and EPICS IOC's allows for easy test development of new hardware interface.
- Write new EPICS IOC hardware interface
- Write FS integration
- Test new hardware in simulated version of control system environment before taking to ATF2.

FS Development

- Improve Server-Client and Client Server-PLACET (etc) socket code
  - Only ascii communication at moment
    - Faster for large data array transfer to implement floats etc
  - Better error reporting and handling
- Handling Multiple Clients (“test installations”)
  - Currently 1 Client supported at a time
  - Socket code developed in a way to support multiple clients if required...
- Parallel processing
  - Core software very serial in nature currently
  - Through additional toolboxes, Matlab supports parallel constructs...
- Improved simulator functionality.
- Improved documentation and examples.
- Improved software management and installation procedure.

Support for Other Codes

- Currently application development active with PLACET
- DR Codes?
  - DR elements exist in FS
  - MAD, SAD, AT, LOCO.... ?
  - Useful for ring matching into EXT for a consistent model etc...
  - Useful for generic fast matching procedures.
- Use combined strengths of different codes in single environment.

Current Software List

- Lucretia + Floodland
- Matlab
- LabCA
- EPICS base 3.14.8
- AsynDriver4-7
- ether_ip
- IOC-qmov
- IOC-ps
- IOC-atf
- PLACET
- Octave
- tcl/tk
- Lucretia2AML
- UAP/AML

Software Location / Installation

- Software for FS exists in 2 CVS repositories at SLAC
- Core FS software (Floodland) is part of Lucretia software repository
  - /afs/slac.stanford.edu/g/ilc/cvs/Lucretia
- Control-software and some ATF integrated simulation software (for Lucretia) in another repo
  - /afs/slac.stanford.edu/g/ilc/cvs/ATF2
  - e.g. Base EPICS software + IOC's etc
- Lucretia comes with binaries required
  - Build instructions also available from Lucretia website (as well as Lucretia simulation documentation etc)
    - http://www.slac.stanford.edu/accel/ilc/codes/Lucretia/
  - Special instructions exist for Lucretia2AML- see relevent directory
- EPICS related software needs building for each installation
  - Uses EPICS build-system: edit and run make script in epics directory

Documentation

- Documentation on SLAC ATF wiki:
  - https://confluence.slac.stanford.edu/display/ATF/ATF2+Flight+Simulator
  - Signup for account from link at root page
  - https://jira.slac.stanford.edu/signup/
- FS Installation instructions
- Matlab/Lucretia use and development notes
- 3rd Party Accelerator code software use notes
- Application documentation should also go here