High Level Control Room Applications Software at SNS

Andrei Shishlo

Accelerator Physics Team Member

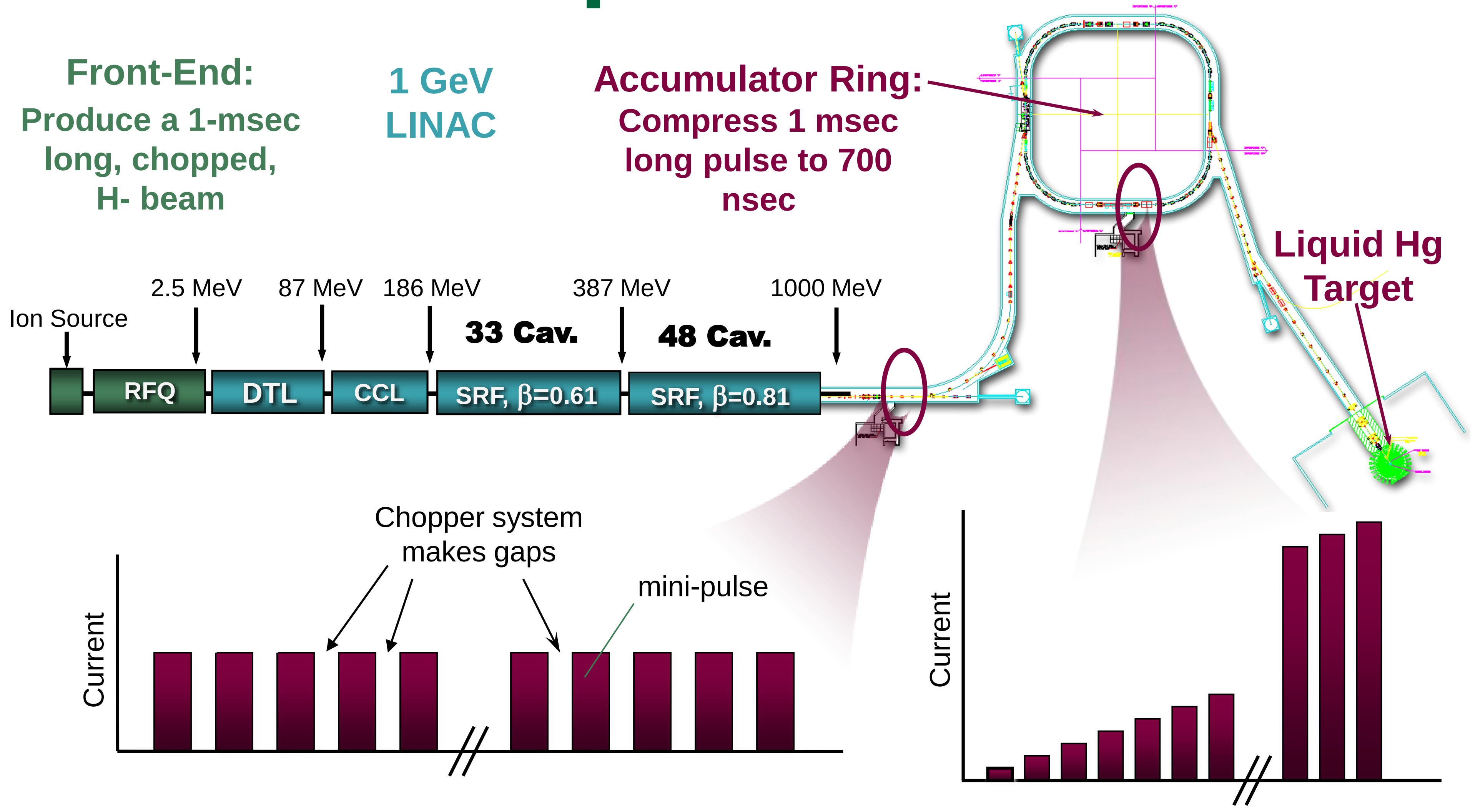
SNS, Oak Ride

August 31, 2018

Outline

- **SNS Accelerator**
- **SNS Partnership**
- **Beginning of Application Programming/ Commissioning**
- **XAL/OpenXAL: Structure / Online Model / Lesson Learned**
- **SCL Tuning: History / Today**

SNS Accelerator Complex



The SNS Partnership

ORNL

Accelerator Systems Division was responsible for integration, installation, commissioning and operation

Application Software Team (initially):

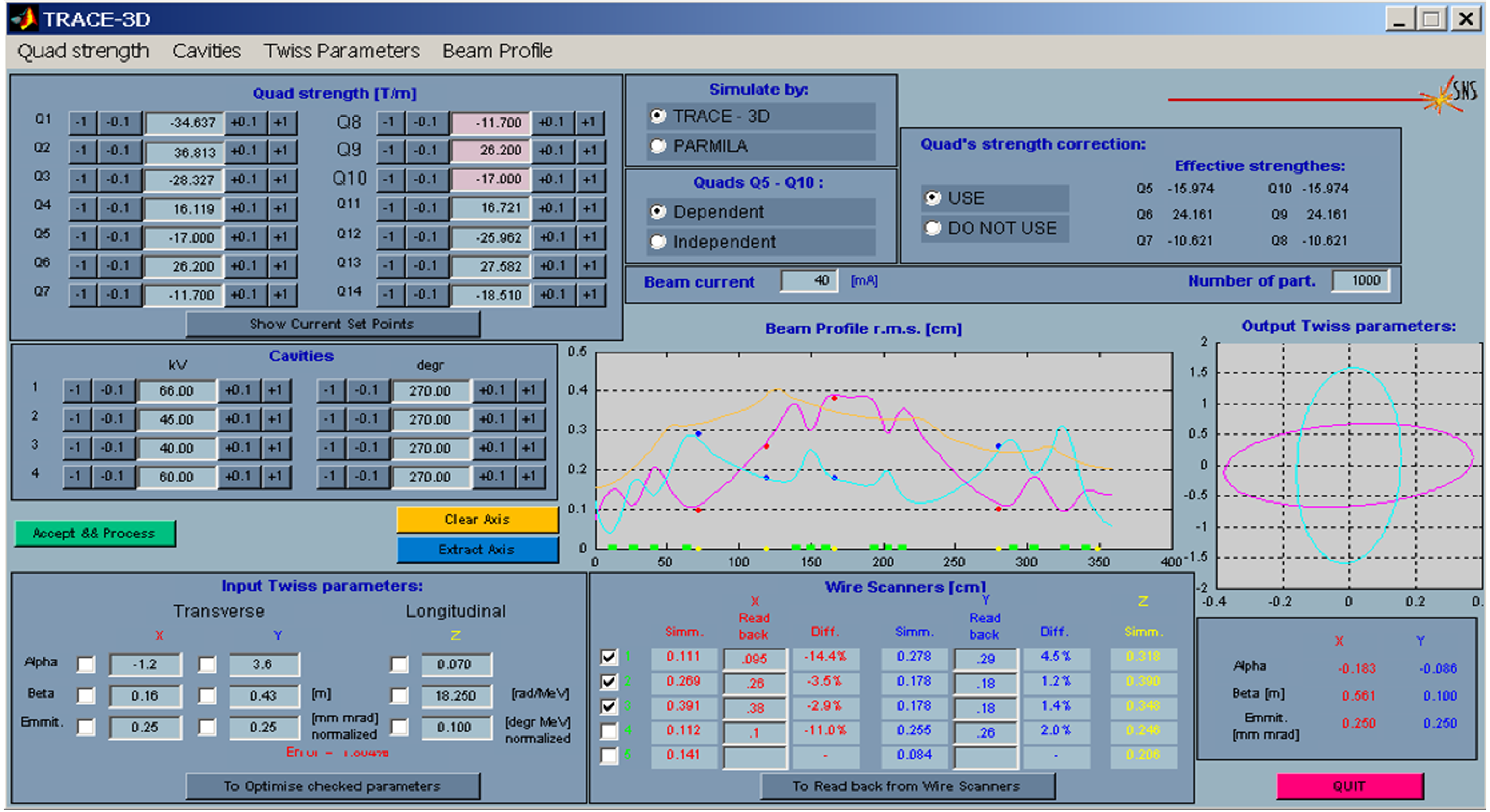
1. SNS – XAL environment and applications (4 FTE)
2. Los Alamos – Online Model (2 FTE)

Application Programming Beginning (2000-2002)

- The different technologies were reviewed: FORTRAN applications, MATLAB, SDDS (Self Describing Data Sets, Argonne), CDev (Jefferson Lab), Java
- Java Advantages:
  - simple, stable
  - object oriented,
  - it runs everywhere,
  - GUI (swing based, now have to move to FX),
  - database interaction,
  - client/server application,
  - Java interface to EPICS CA existed,
  - appeal to young physicist/developers
- Java Disadvantages (at that time)
  - Graphics (contours, error bars, real-time, 3-D, …),
  - Open source mathematical libraries less mature,
  - Most AP members used MatLab
- Application programming requirements was formulated, a list of programs was constructed, manpower needed is 43 FTE (Full Time Equivalent) for 3.5 years of commissioning, accelerator physics, controls, and diagnostics groups are involved
- Two versions of applications: for commissioning and for operations. Commissioning versions are streamlined applications with minimal user interface
- The Application Programming Team (John Galambos - leader) was created inside Accelerator Physics Group to start development of Java infrastructure and high level physics applications

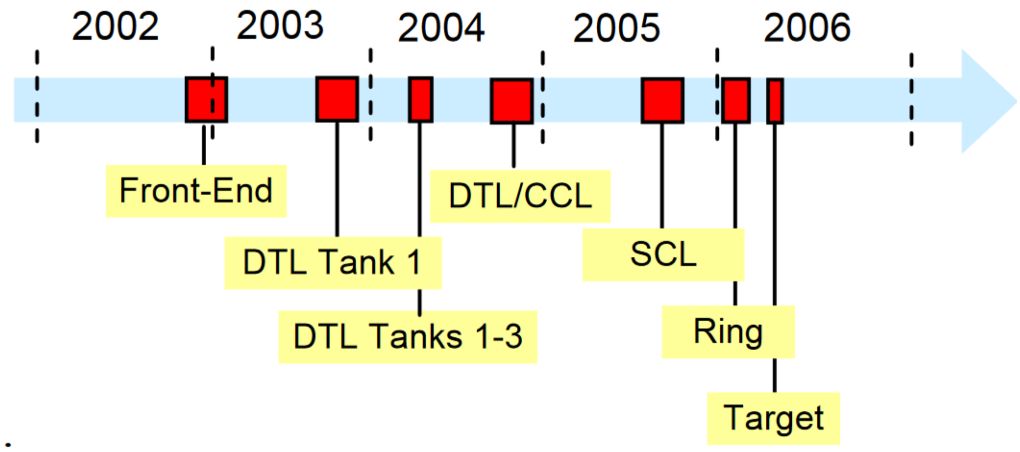
MEBT and DTL commissioning: MatLab prototypes of some of applications were written first by AccPhy group members, and then they were rewritten in Java to insure a successful commissioning

MEBT Optics MatLab App based on Trace3D and Parmila

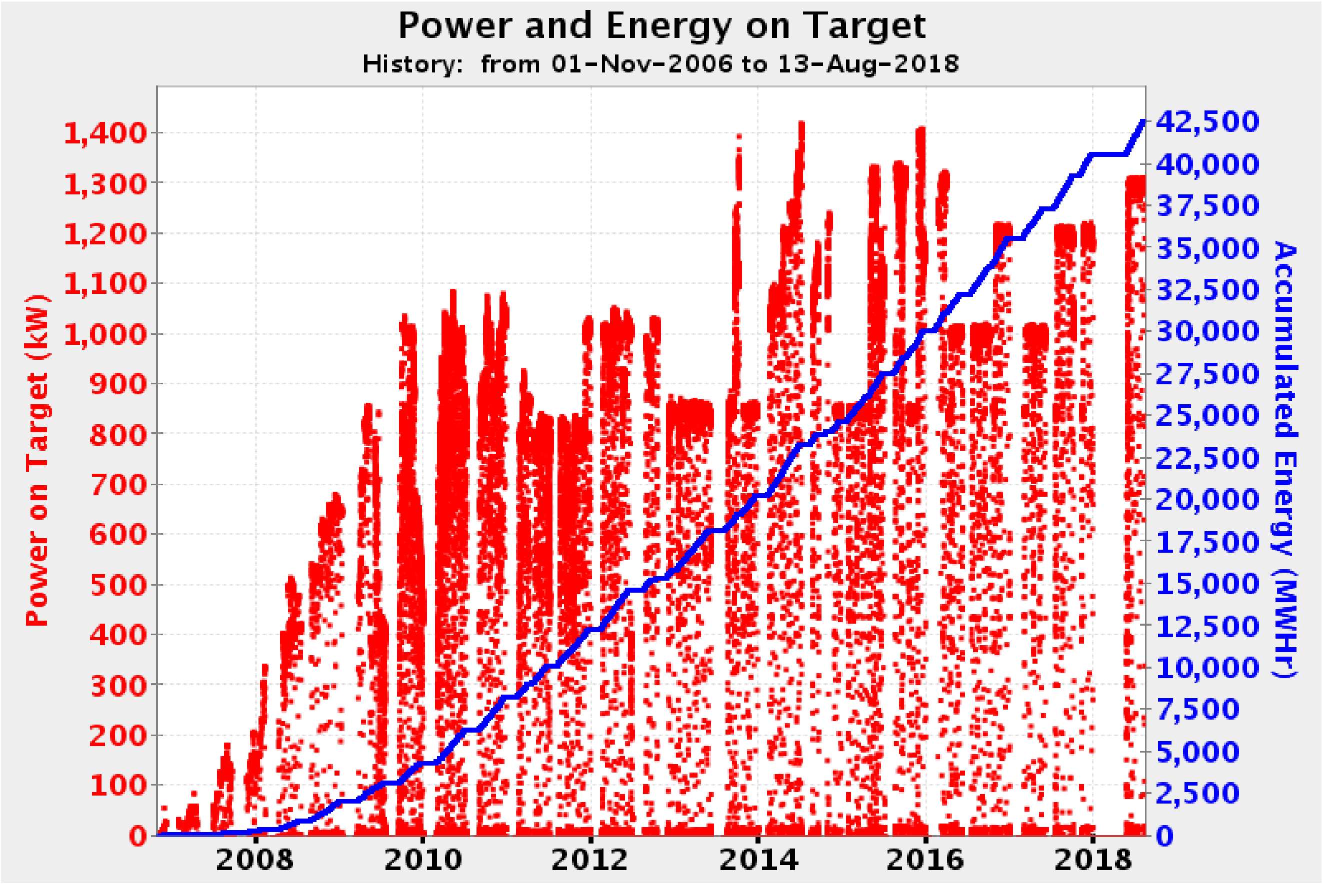


(Courtesy of A. Aleksandrov)

SNS Commissioning and Power on Target History



SNS Beam Commissioning Schedule



- Commissioning was squeezed between Installation activities.
- Try-and-learn iterations approach to software applications development
- Much less time was available for beam commissioning than originally planned.
- Pace of commissioning accelerated at the end

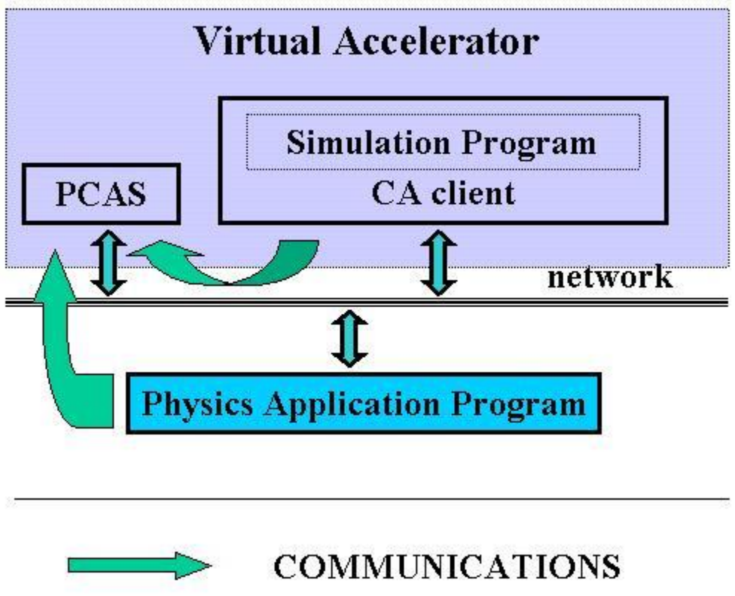
Lessons after First Step

- Need to familiarize people with application features before commissioning.
- Need GUI interfaced applications for general users.
- Have integrated help capability, common look/feel
- Testing with Virtual Accelerator before commissioning helped

Actions:

- The practice of live lessons for applications become a common practice
- The development of the Application Framework initiated
- Proceed with the Virtual Accelerator development

Virtual Accelerators

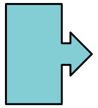


"Virtual accelerator" is a model imitating the real machine. In the case of EPICS data exchange It looks like a real machine from the EPICS channel access view, because operates with real process variable (PV) names, and produces a reasonable response generated by the simulation model.

- **PCAS - Portable Channel Access Server**
- **Simulation Program – Accelerator Model**
- **CA client – Interface to the Simulation Program + channel access client**
- **Physics Application Program – application under development**

Simulation Program:

- Trace3D
- PARMILA
- XAL Online Model

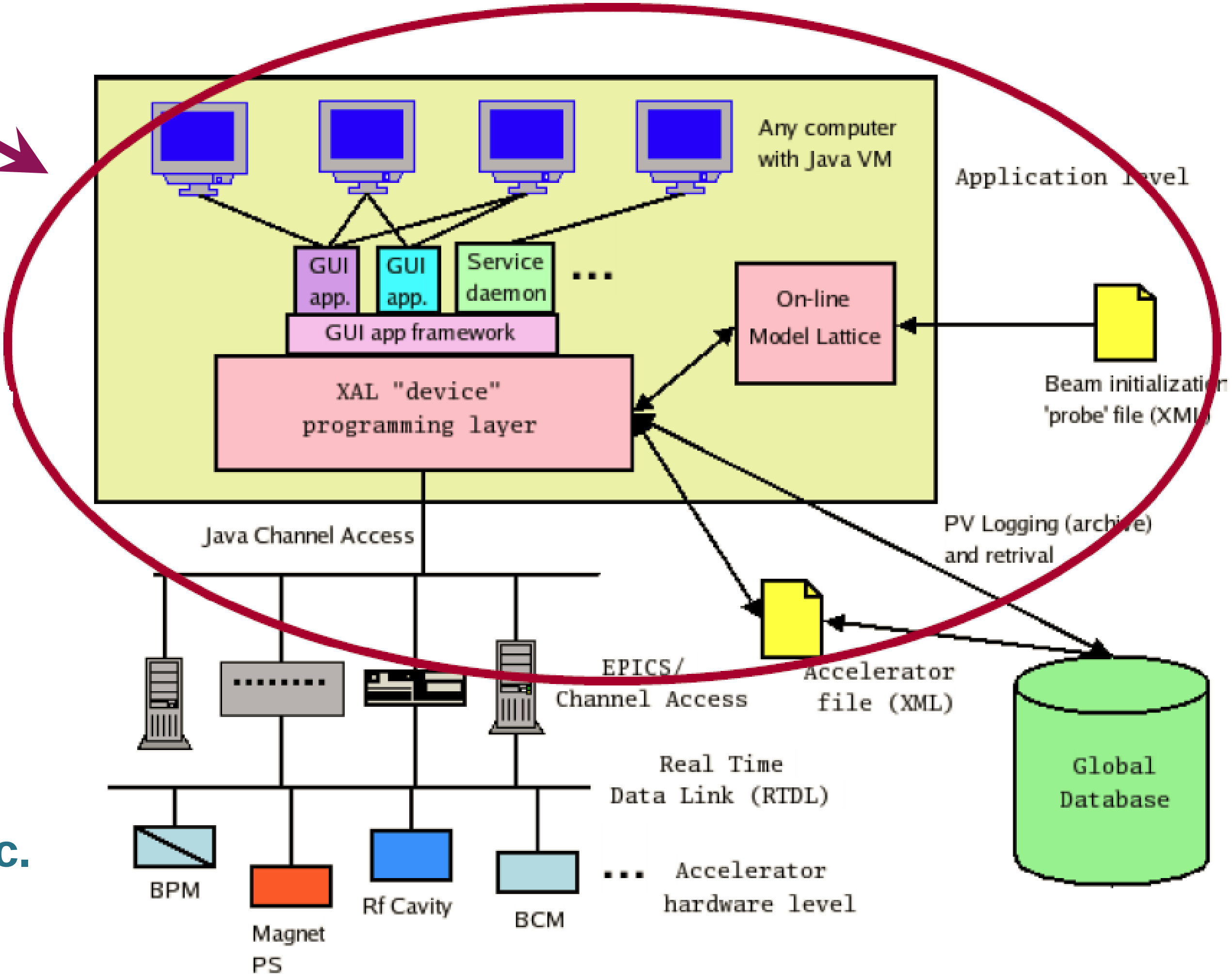


The main programs were modified. No system calls or output files' analysis.

Now it is an XAL Application.
Very useful on early stages and for demonstrations.

XAL/OpenXAL Structure

XAL and
its apps



SNS application software environment for:

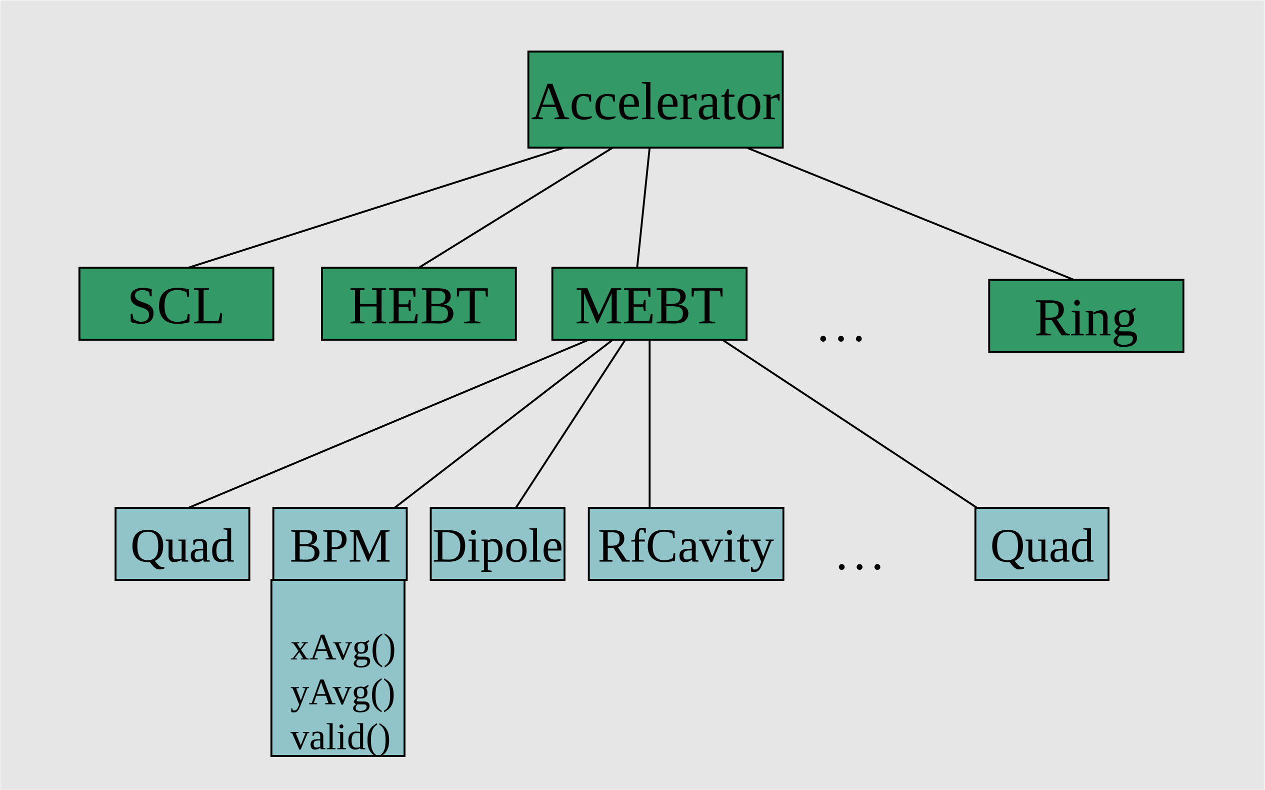
High level physics application

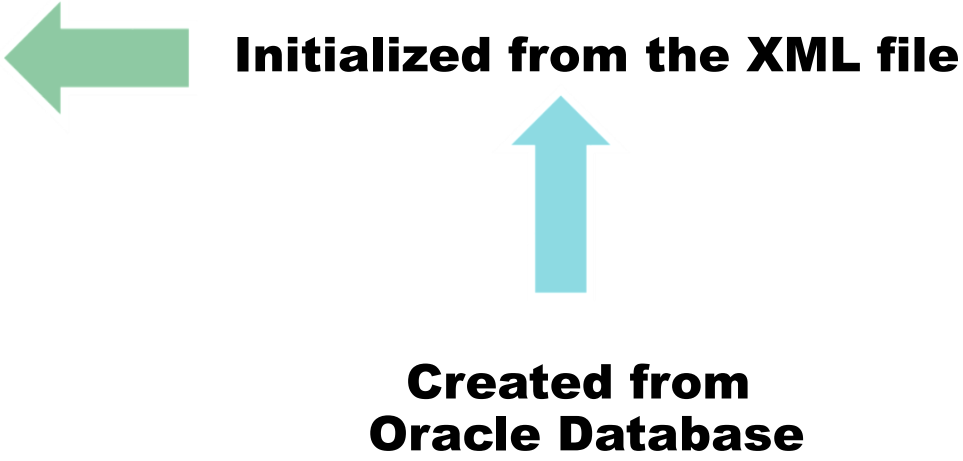
For modeling operation and accelerator physics studies

XAL includes:

- Online Model
- Tool Box (math, optimization, plotting etc. packages)
- Application Framework
- Services
- Abstract CA Clients layer

XAL/OpenXAL -Accelerator Class- Hierarchical “Device” View





- Java class structure that provides a hierarchical “device” view of the accelerator to the application programmers
- It knows everything that is important about any lattice device
- Setup from database through XML file, EPICS connections hidden

Online Model

- Envelop and Single Particle Dynamics, inherited from Trace-3D and PARMILA
- Simulates charged particle dynamics through specified accelerator sequences
- Supports both linear sequences and rings
- Calculates Twiss parameters, transport matrices, energy and orbit distortions
- Six dimensional phase space propagation
- Includes space charge forces for envelop propagation
- Optics synchronization:
  - design
  - live machine
  - PV Logger snapshot
  - custom values (or combination of these sources)
- Fast enough to use inside optimization tasks in the interactive mode

Is not suitable for FRIB – no multiple charge states

Models Used at SNS

XAL Online Model	A part of XAL/OpenXAL programming framework. Envelop and Single Particle Dynamics, inherited from Trace-3D and PARMILA
PARMILA	It was used for the SNS linac design. PARMILA was occasionally used as an online tool for matching the beam into DTL and CCL (under MATLAB GUI script) and for offline analysis.
IMPACT	It is a parallel computer PIC accelerator code which includes 3D space charge calculations. At SNS it was used for offline analysis.
Trace-3D	Envelope Model. The algorithms were migrated to XAL online model. It was used for benchmarks.

Now we are using PyORBIT code for offline simulations

Lesson Learned from XAL/OpenXAL development

What we did right:

- **Early staged commissioning approach**
- **Iterative Approach for Commissioning Tools**
- **Using physicists (i.e. commissioners) to write applications (Need a core group of "mentor" programmers)**
- **Educational efforts**

In XAL Development:

- **Choose Java**
- **Initialization files created from a database**
- **Online Model**
- **Application Framework**
- **Scripting (Jython/Ruby)**

What we did wrong:

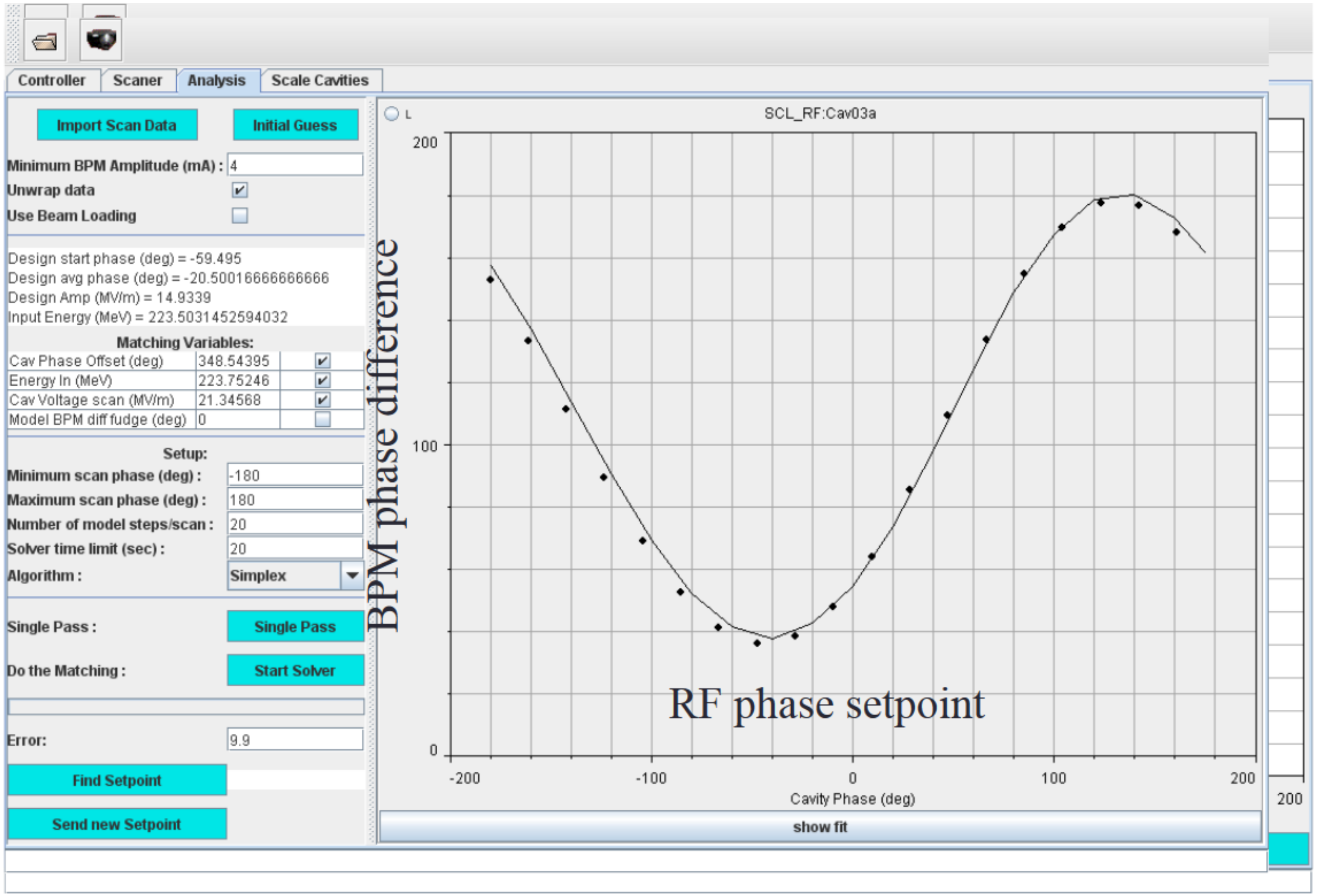
- **Most applications and some of tools are SNS specific**
- **Lack of documentation**
- **Save/Restore App uses only a database**
- **Did not implement service daemons to reduce EPICS traffic**
- **We used commercial plotting package (JClass) in the open source software (XAL)**

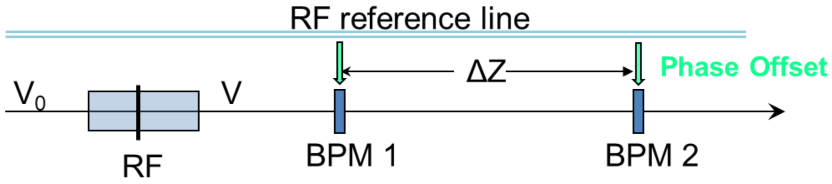
List of Useful Apps for Commissioning

App Name	Description	App Name	Description
SCORE	Save/Compare/Restore: to snapshot the state of the accelerator for fast recovery. Originally: data base only. Now: data base + XML file	PVLogger	The service that other apps can use to create a log of machine parameters for an offline model initialization
Orbit Correction	Save trajectories, restore to 0 or to saved trajectory.	PTA	Profile data and Tools Analysis: transverse profile measurements
Beam Loss Viewer	To see, save, compare to snapshot, compare to limits, …	Transverse Matching	
General Purpose Scan	Scan one or two parameters and measure an arbitrary number of parameters. Basic operations with data	Beam Arrival Time Snapshot	Snapshot of BPM phases along the linac (save, compare, etc)
Knobs	Local transverse bump at an arbitrary point with arbitrary of correctors	General Beam Model	Beam model with GUI stable from machine
Energy Meter	Display real time energy measurement from TOF	Rescale SCL Cavities	Retuning SCL linac
Trajectory difference	Apply transverse kick and compare model and measured change in the trajectory.	SCL Cavities Setup	Scan SCL cavities to setup synchronous phases

SCL Tuning XAL/OpenXAL Application

SLACS – Superconducting Linac Automated Cavity Setter (XAL Application, J. Galambos)





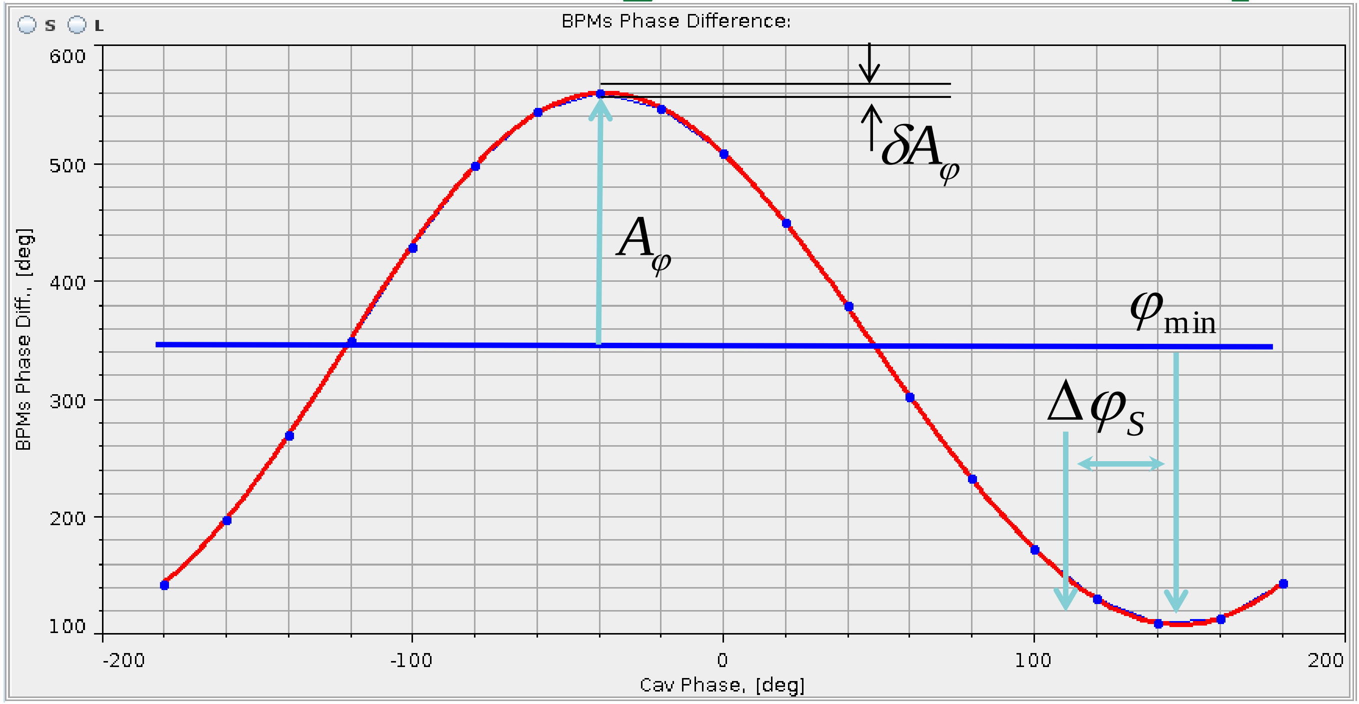
- Choose two BPMs
- All cavities between them are OFF
- Perform RF phase scan
- Find minimal phase diff. (max energy gain) and subtract 18^0
- Energy from TOF for two BPMs

SCL Tune-Up Time:

Dec. 2005:	101 hrs	E_{out} = 925 MeV
July 2006:	57 hrs	E_{out} = 855 MeV
Oct 2006:	30 hrs	E_{out} = 905 MeV
Jan. 2007:	6 hrs	E_{out} = 905 MeV

Even 6 hours is too long! And we wanted to improve accuracy.

SCL RF Cavity Phase Setup - Errors



We do not need time-calibrated BPMs!

$$\delta\varphi_{\min} \approx \frac{1}{\sqrt{N}} \frac{\delta\phi_{BPM}}{A_\varphi}$$

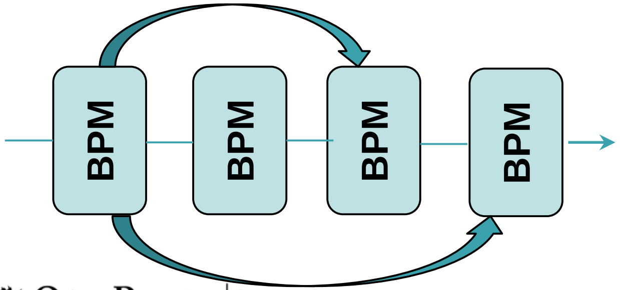
$$A_\varphi \approx \Delta z_{BPM} \cdot \frac{1}{(\gamma \cdot \beta)^3} \cdot E_0 TL$$

Conclusions

- **Two neighbor BPMs – worst case**
- **More energy – less accurate the RF phase**
- **Smaller step – 1/square effective**

We want to use BPMs as far as possible!

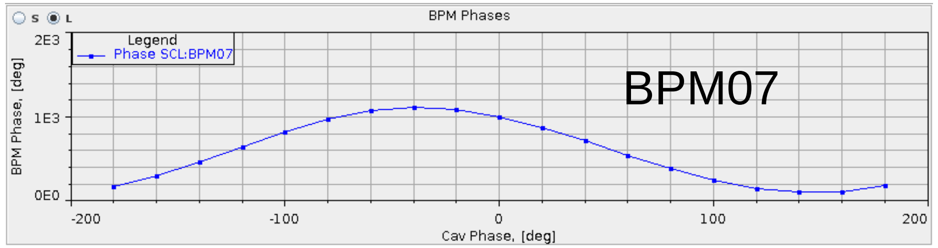
Less steps (N) – faster the scan!

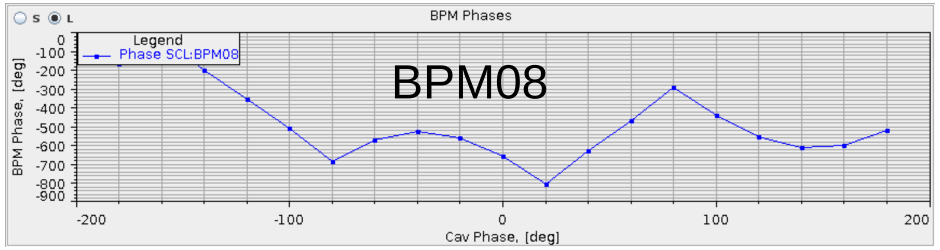


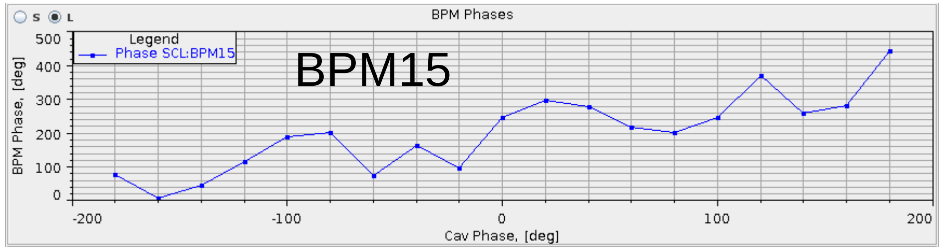
A “Big Phase Step” Problem

- BPMs measure phase in -180^0 to +180^0 range
- To get sinusoidal curve we have to unwrap the phase scan
- Usually, we do this by using the previous phase point of the scan
- Therefore we have to use small steps to avoid more that 180^0 gain in one step
- If we use far away BPM pairs, it could be problem for the “big phase step”

$$A_{\varphi} \cdot \sin(\Delta\phi_{RF}) < 180^{0}$$







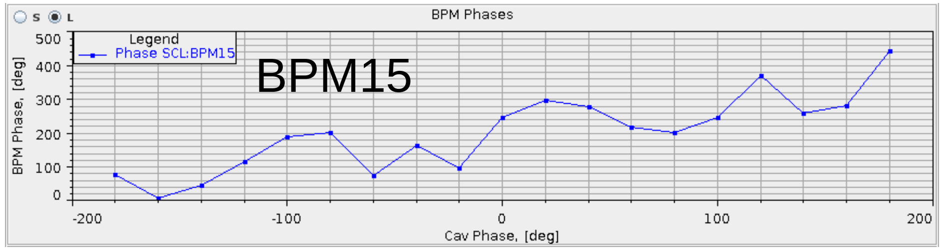
An example of Cav01a scan.
We cannot go further BPM07 with the step size 20^0

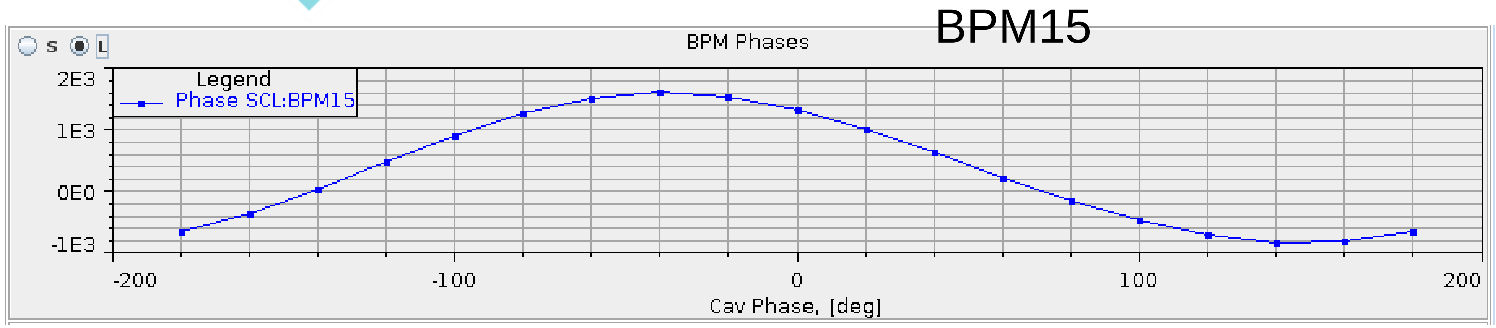
Solution for the "Big Step" Problem

Most simple – iterative approach – the unwrapping is done by using not only the previous point, but also the previous and current points from the previous BPM. The iteration starts with the BPM closest to the cavity.

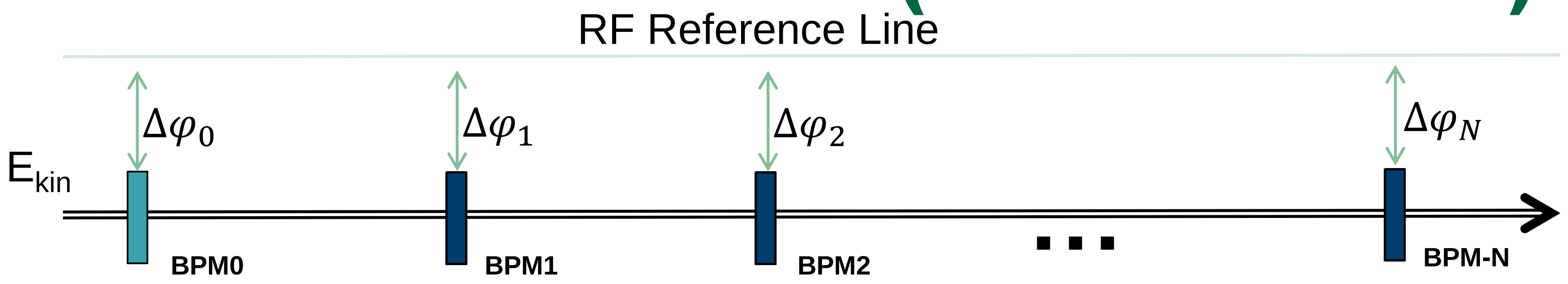
$$(A_{\varphi}^{(k)} - A_{\varphi}^{(k-1)}) \cdot \sin(\Delta\phi_{RF}) < 180^{0}$$

Phase step size can be 40^0, 60^0 or may be even 90^0.
It means 10-15 minutes scan for the whole SCL.
In reality, we limit ourselves by 30-40 mins.





BPM Phase Offsets in HEBT1 (Strait Section)



$$\varphi_i = 360 \cdot f_{BPM} \frac{s_i}{\beta \cdot c} + \Delta\varphi_i$$

1. We do not know the phase offsets (BPMs are not calibrated).
2. We can find them if we know the energy (beta) and BPM phases
3. We use the ring to measure the energy
4. One of the BPMs is a reference BPM

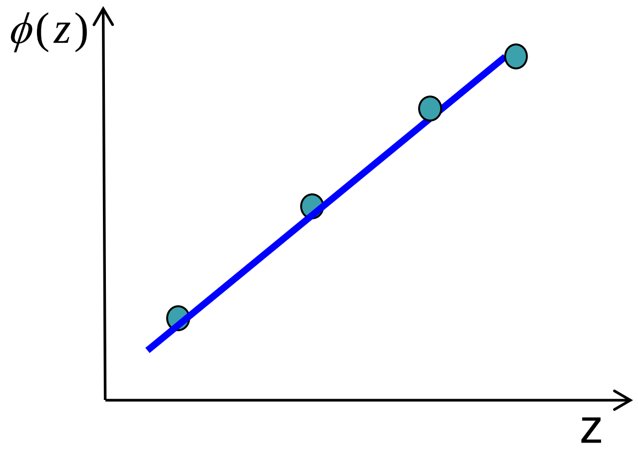
After we know the BPM phase offsets, we can calculate the energy by reverting the formula

BPM Phase Offsets Calibration 2014.02.04

SCL:BPM00a	0.0 +- 0.1
SCL:BPM00b	-0.0 +- 0.1
SCL:BPM01	-173.2 +- 0.1
SCL:BPM02	-171.8 +- 0.1
SCL:BPM03	-168.2 +- 0.1
SCL:BPM04	-103.2 +- 0.1
SCL:BPM05	-138.1 +- 0.1
SCL:BPM06	-138.4 +- 0.1
SCL:BPM07	-134.9 +- 0.1
SCL:BPM08	-135.3 +- 0.2
SCL:BPM09	45.3 +- 0.2
SCL:BPM10	48.7 +- 0.2
SCL:BPM11	53.6 +- 0.2
SCL:BPM12	44.7 +- 0.2
SCL:BPM13	-136.3 +- 0.2
SCL:BPM14	-141.2 +- 0.3
SCL:BPM15	7.9 +- 0.3
SCL:BPM16	9.7 +- 0.3

SCL:BPM17	-47.3 +- 0.3
SCL:BPM18	127.7 +- 0.3
SCL:BPM19	122.4 +- 0.4
SCL:BPM20	134.2 +- 0.4
SCL:BPM21	145.5 +- 0.0
SCL:BPM23	56.9 +- 0.9
SCL:BPM24	52.7 +- 1.1
SCL:BPM25	52.6 +- 0.5
SCL:BPM26	66.9 +- 0.5
SCL:BPM27	-119.5 +- 0.8
SCL:BPM28	
SCL:BPM29	-144.9 +- 1.4
SCL:BPM30	-146.0 +- 1.7
SCL:BPM31	42.0 +- 0.7
SCL:BPM32	43.2 +- 0.5
HEBT:BPM01	
HEBT:BPM02	-15.8 +- 0.4
HEBT:BPM03	79.5 +- 0.3
HEBT:BPM04	57.7 +- 0.5
HEBT:BPM05	64.3 +- 0.1
HEBT:BPM06	44.8 +- 0.2
HEBT:BPM08	74.6 +- 0.8
HEBT:BPM10	-104.4 +- 0.4
HEBT:BPM11	72.4 +- 0.4

We clearly see the calibrated pairs of BPMs.



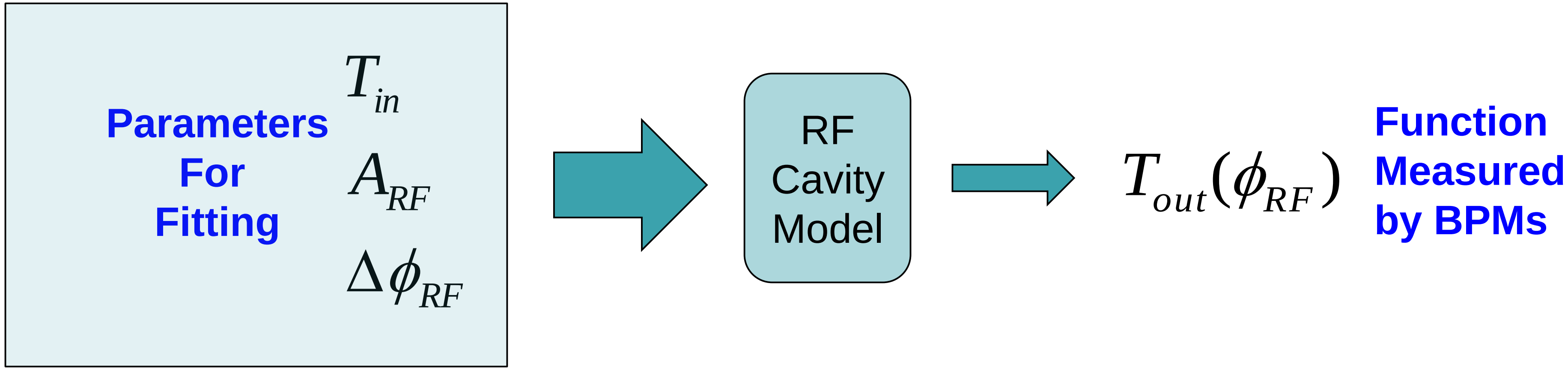
$$\phi(z) = \phi_0 + \omega \cdot \frac{z}{c} \cdot \left(\frac{1}{\beta}\right)$$

Now energy is found by using all available BPMs that are downstream of the scanned cavity.

$$\delta T = \frac{c \cdot m}{\Delta z \cdot \omega} \cdot (\gamma \cdot \beta)^3 \cdot \delta\phi_{BPM}$$

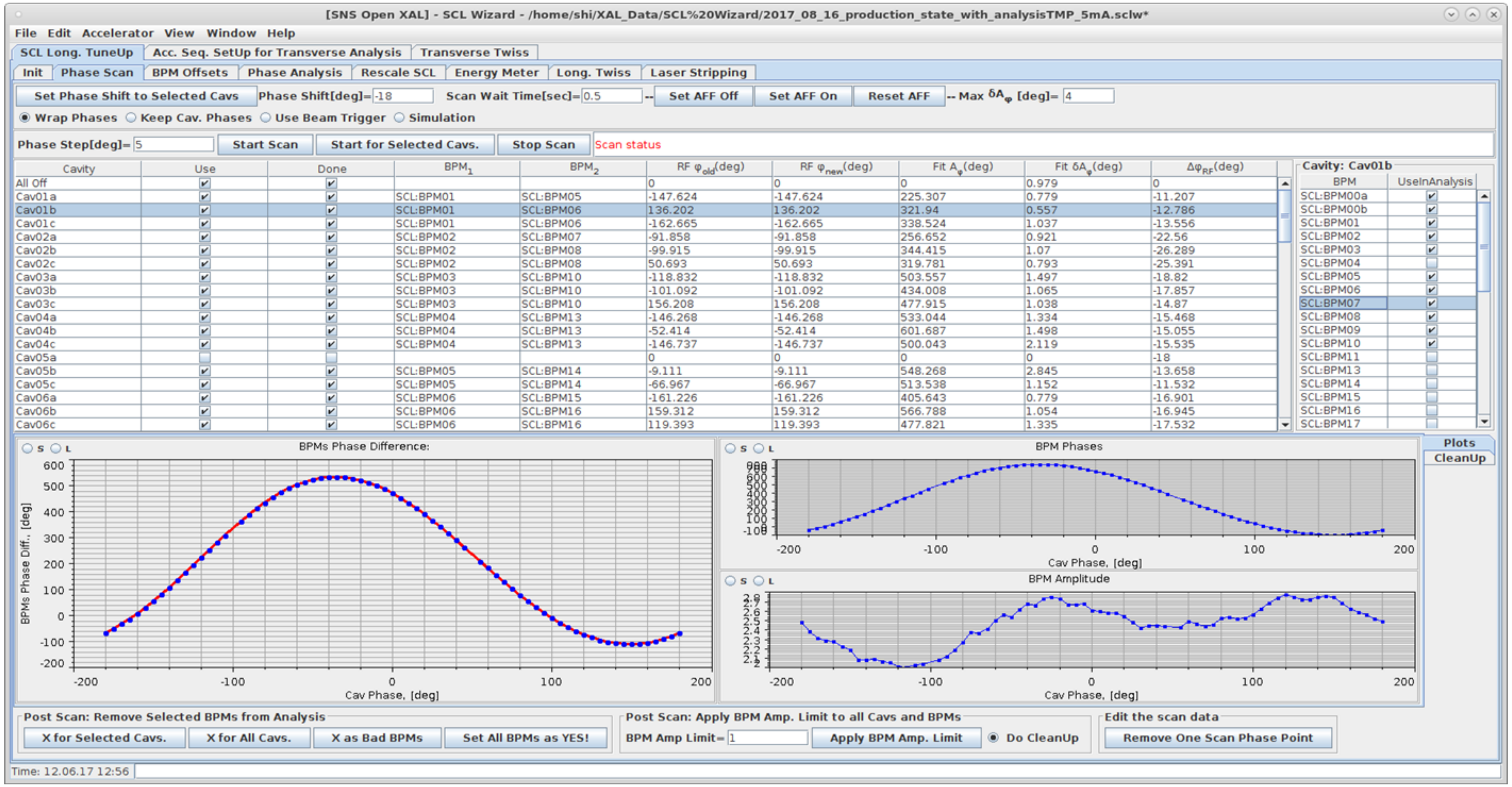
Accuracy of the energy calculation from two BPMs

Model Based Phase Scan Analysis



- **After using the SNS ring for BPMs' calibration we know the energy for each phase point of each cavity**
- **We fit the measured kinetic energy vs. cavity phase by using the input energy, the cavity amplitude, and the cavity phase offset.**
- **We use XAL Online Model**
- **The input energy for one cavity is not the output energy of the previous one. The difference shows the model imperfections.**

SCL Tuning Wizard



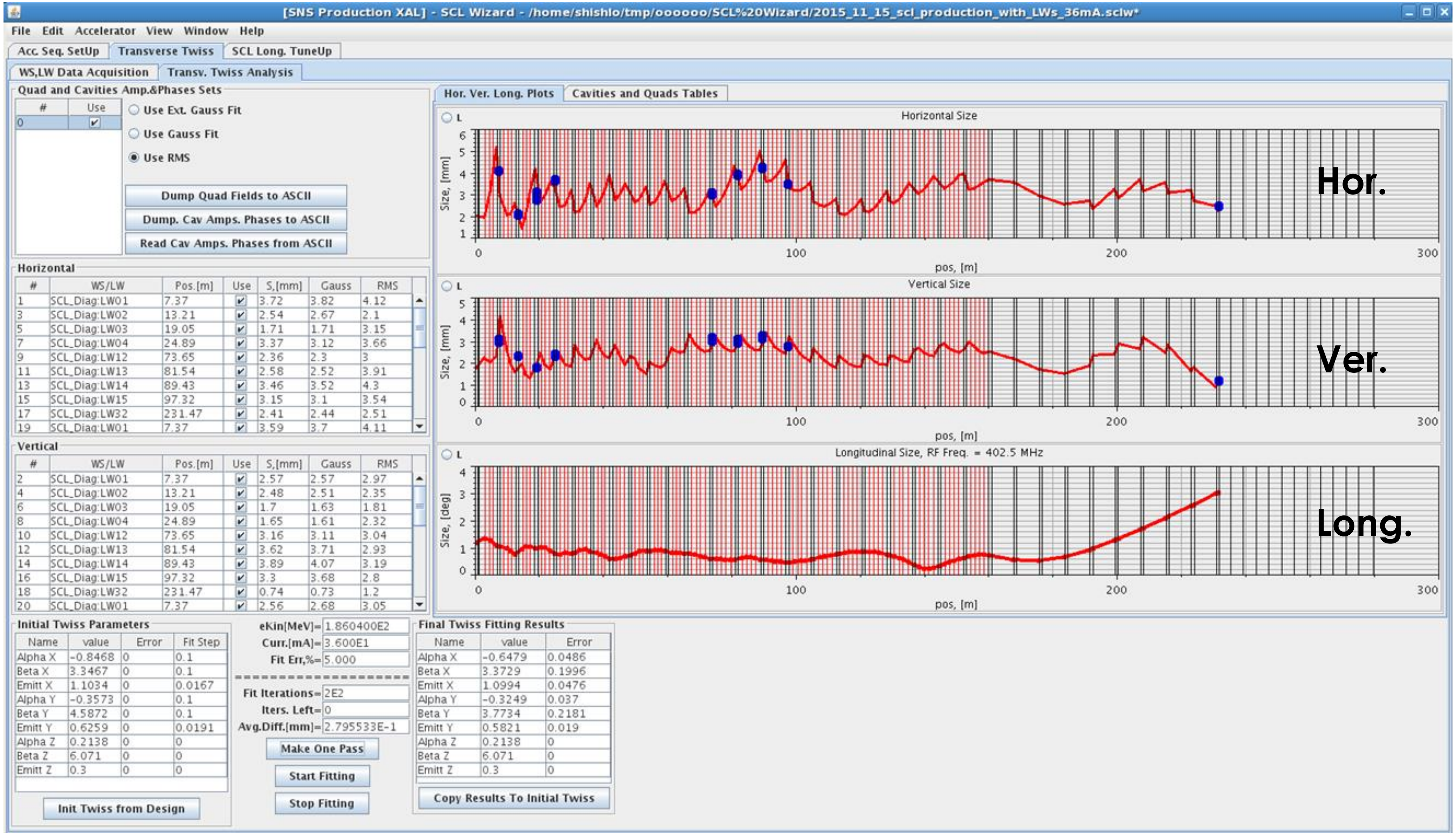
SCL Tuner Wizard:

- Phase Scan
- BPM Phase Offsets calculations
- Model Based Analysis
- Model Based Rescale
- Energy Meter
- Longitudinal Twiss Analysis
- Transverse Twiss Analysis

Implementation: Jython + OpenXAL

- "One button" tune up application
- Takes about 40 min to tune the whole SCL with 81 cavities
- The SCL can be retuned instantly if we have to switch off one of the cavities
- Non-destructive scan capability

SCL Tuner Wizard: Transverse RMS Sizes, Matching



Lessons Learned from Latest Tuning Apps Development

- Tuning automation is a must
- Integrated applications (wizards) are more powerful
- These apps are not only for tuning, but they are also for studies
- Iterative approach is forever