

# High Level Control Room Applications Software at SNS

**Andrei Shishlo**

**Accelerator Physics Team Member**

**SNS, Oak Ridge**

**August 31, 2018**

# Outline

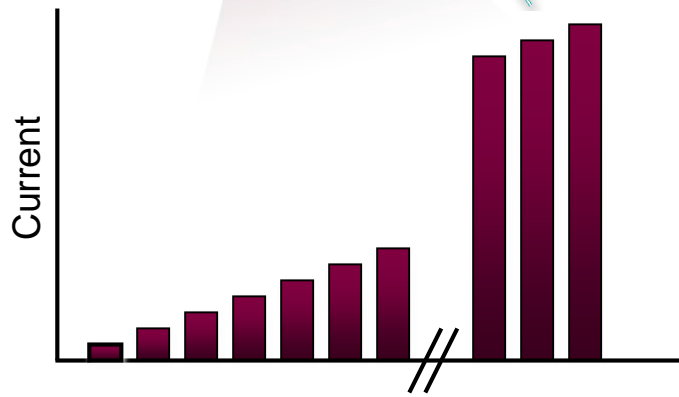
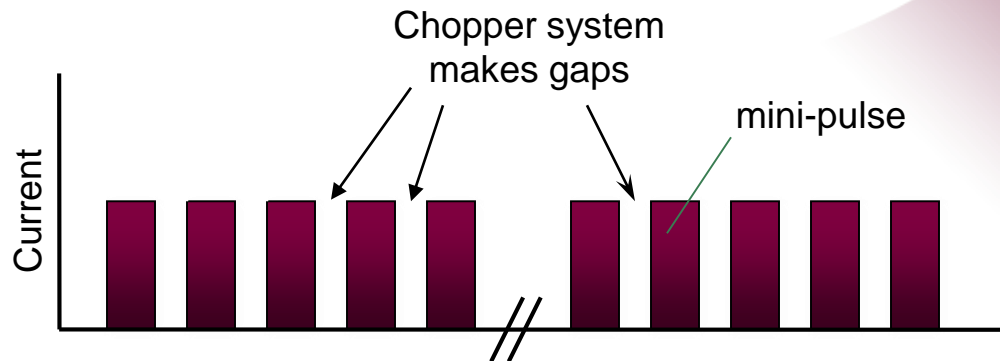
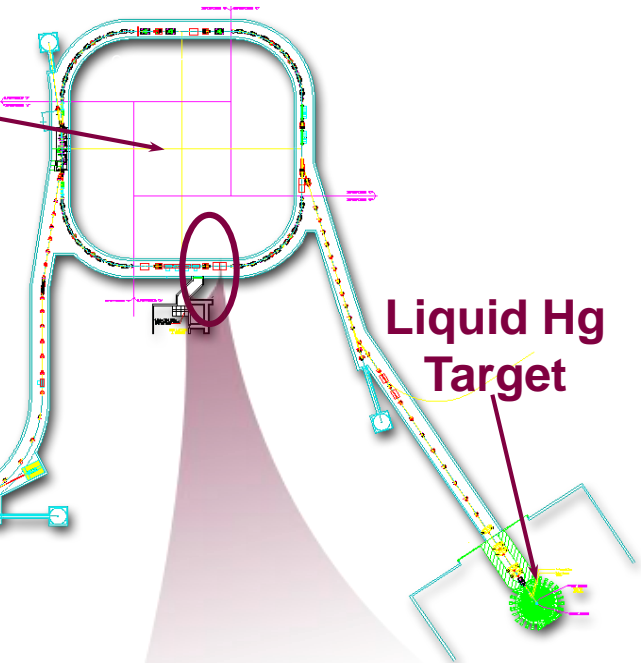
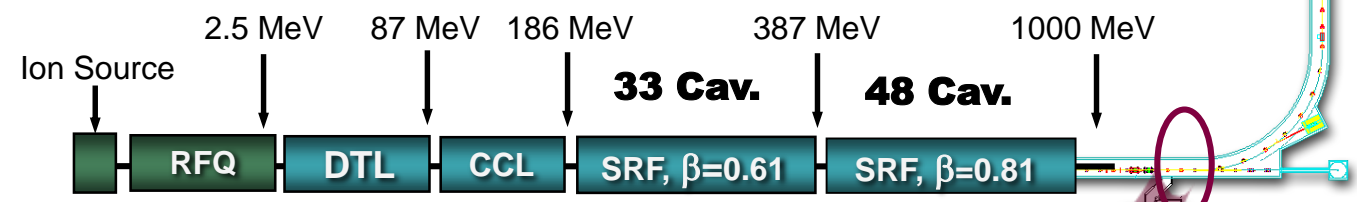
- **SNS Accelerator**
- **SNS Partnership**
- **Beginning of Application Programming/ Commissioning**
- **XAL/OpenXAL: Structure / Online Model / Lesson Learned**
- **SCL Tuning: History / Today**

# SNS Accelerator Complex

**Front-End:**  
Produce a 1-msec long, chopped, H- beam

**1 GeV LINAC**

**Accumulator Ring:**  
Compress 1 msec long pulse to 700 nsec



# 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

**TRACE-3D**

Quad strength Cavitys Twiss Parameters Beam Profile

**Quad strength [T/m]**

Q1	-1	-0.1	-34.637	+0.1	+1	Q8	-1	-0.1	-11.700	+0.1	+1
Q2	-1	-0.1	36.813	+0.1	+1	Q9	-1	-0.1	26.200	+0.1	+1
Q3	-1	-0.1	-28.327	+0.1	+1	Q10	-1	-0.1	-17.000	+0.1	+1
Q4	-1	-0.1	16.119	+0.1	+1	Q11	-1	-0.1	16.721	+0.1	+1
Q5	-1	-0.1	-17.000	+0.1	+1	Q12	-1	-0.1	-25.962	+0.1	+1
Q6	-1	-0.1	26.200	+0.1	+1	Q13	-1	-0.1	27.582	+0.1	+1
Q7	-1	-0.1	-11.700	+0.1	+1	Q14	-1	-0.1	-18.510	+0.1	+1

Show Current Set Points

**Simulate by:**

TRACE - 3D  
 PARMILA

**Quads Q5 - Q10 :**

Dependent  
 Independent

**Quad's strength correction:**

USE  
 DO NOT USE

**Effective strengthes:**

Q5	-15.974	Q10	-15.974
Q6	24.161	Q9	24.161
Q7	-10.621	Q8	-10.621

Beam current: 40 [mA]      Number of part.: 1000

**Cavitys**

	kV		degr							
1	-1	-0.1	66.00	+0.1	+1	-1	-0.1	270.00	+0.1	+1
2	-1	-0.1	45.00	+0.1	+1	-1	-0.1	270.00	+0.1	+1
3	-1	-0.1	40.00	+0.1	+1	-1	-0.1	270.00	+0.1	+1
4	-1	-0.1	60.00	+0.1	+1	-1	-0.1	270.00	+0.1	+1

Accept & Process      Clear Axis      Extract Axis

**Beam Profile r.m.s. [cm]**

**Output Twiss parameters:**

**Input Twiss parameters:**

Transverse      Longitudinal

	X	Y	Z
Alpha	<input type="checkbox"/> -1.2	<input type="checkbox"/> 3.6	<input type="checkbox"/> 0.070
Beta	<input type="checkbox"/> 0.16	<input type="checkbox"/> 0.43 [m]	<input type="checkbox"/> 18.250 [rad/MeV]
Emmit.	<input type="checkbox"/> 0.25 [mm mrad] normalized	<input type="checkbox"/> 0.25 [mm mrad] normalized	<input type="checkbox"/> 0.100 [degr/MeV] normalized

Er/L = 1.00478

To Optimise checked parameters

**Wire Scanners [cm]**

	X Simm.	X Read back	Diff.	Y Simm.	Y Read back	Diff.	Z Simm.
<input checked="" type="checkbox"/> 1	0.111	.095	-14.4%	0.278	.29	4.5%	0.318
<input checked="" type="checkbox"/> 2	0.269	.26	-3.5%	0.178	.18	1.2%	0.390
<input checked="" type="checkbox"/> 3	0.391	.38	-2.9%	0.178	.18	1.4%	0.340
<input type="checkbox"/> 4	0.112	.1	-11.0%	0.255	.26	2.0%	0.240
<input type="checkbox"/> 5	0.141		-	0.084		-	0.200

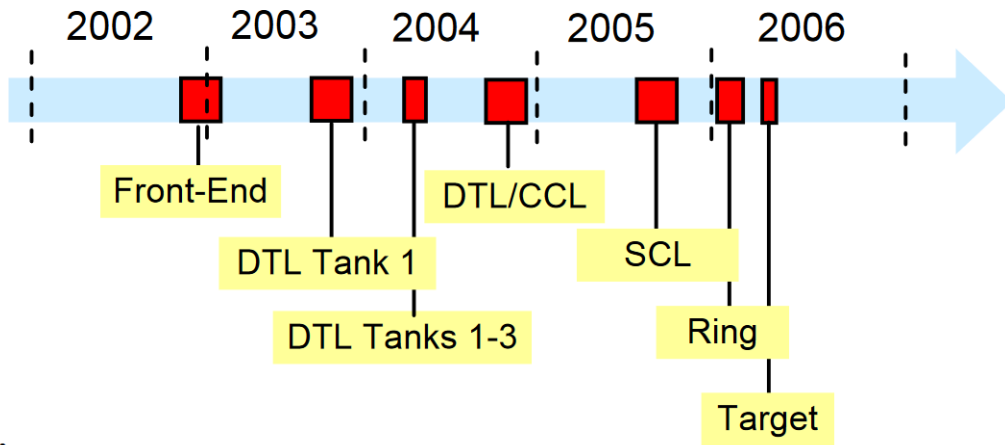
To Read back from Wire Scanners

	X	Y
Alpha	-0.183	-0.086
Beta [m]	0.561	0.100
Emmit. [mm mrad]	0.250	0.250

QUIT

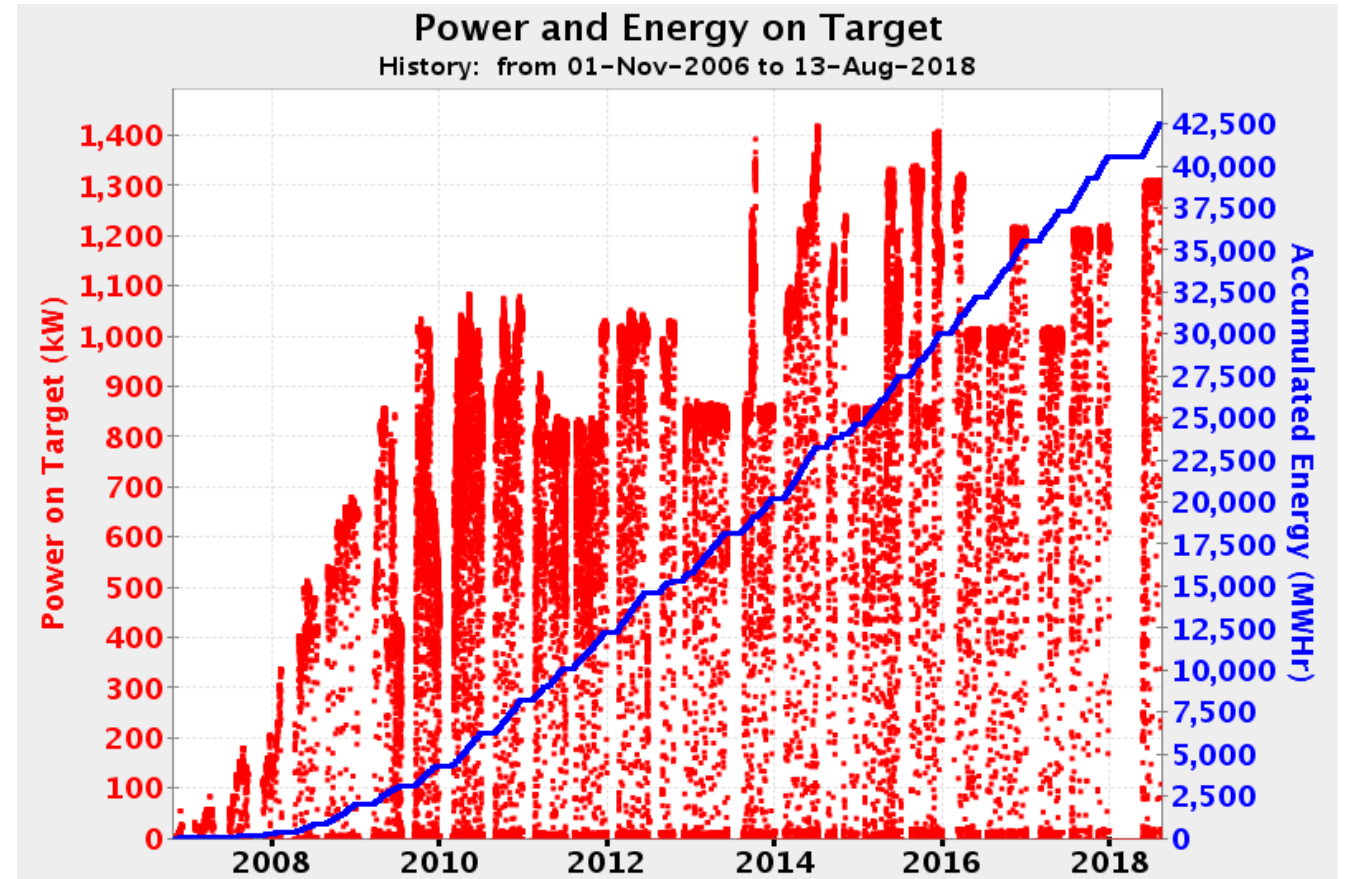
(Courtesy of A. Aleksandrov)

# 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

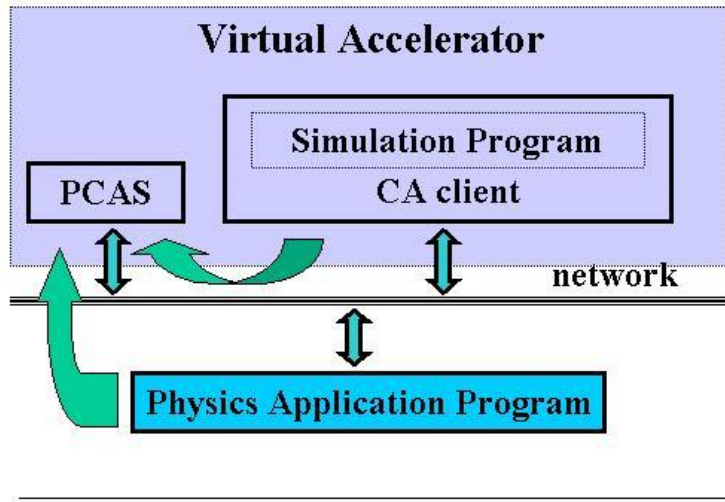
- Need to familiarize people with application features before commissioning.
- Need GUI interfaced applications for general users.
- Have integrated help capability, common look/feel
- Testing with Virtual Accelerator before commissioning helped

## 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

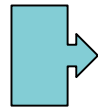


“Virtual accelerator” is a model imitating the real machine. In the case of EPICS data exchange it looks like a real machine from the EPICS channel access view, because it operates with real process variable (PV) names, and produces a reasonable response generated by the simulation model.

- 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

SNS application software environment for:

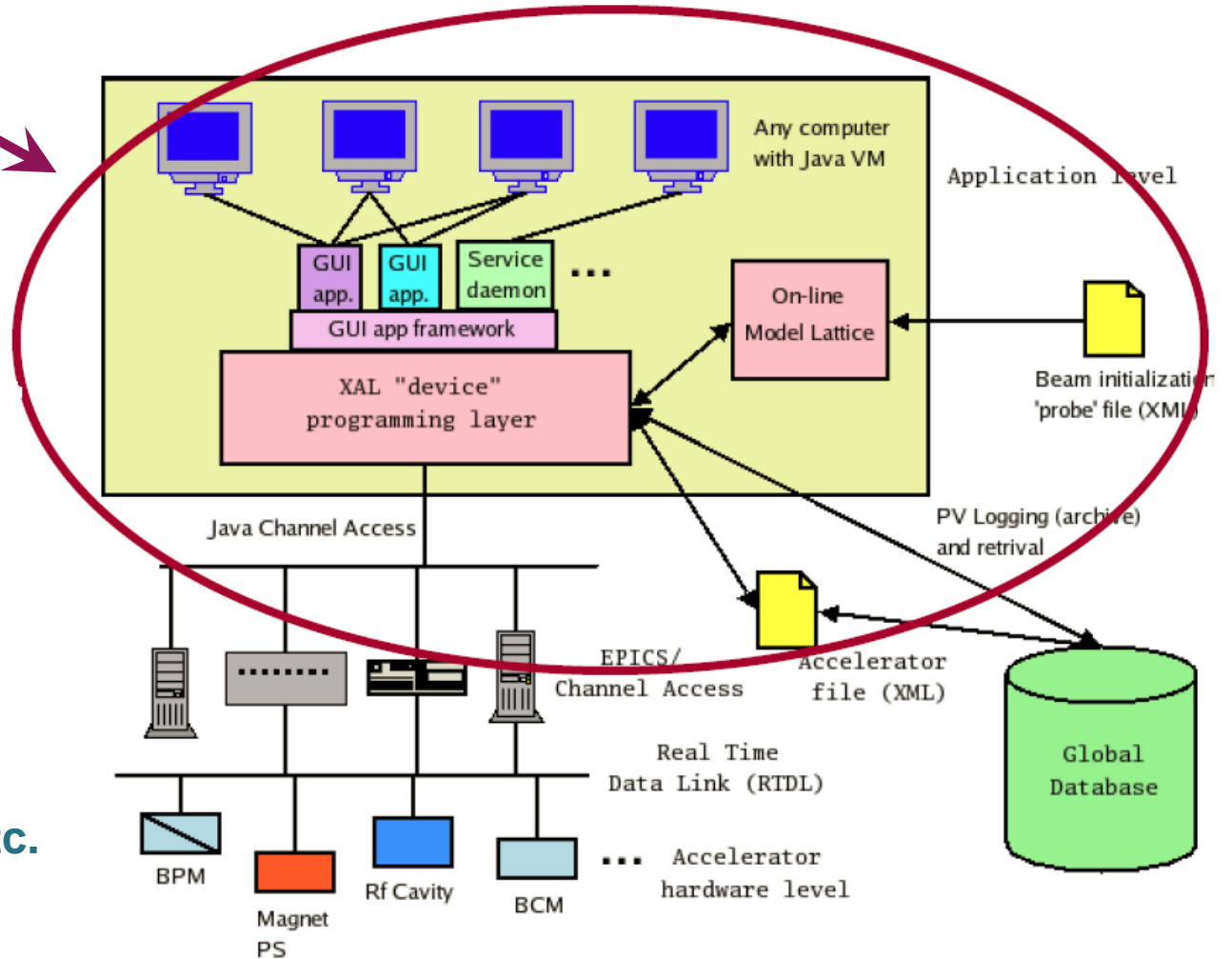
High level physics application

For modeling operation and accelerator physics studies

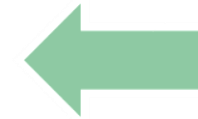
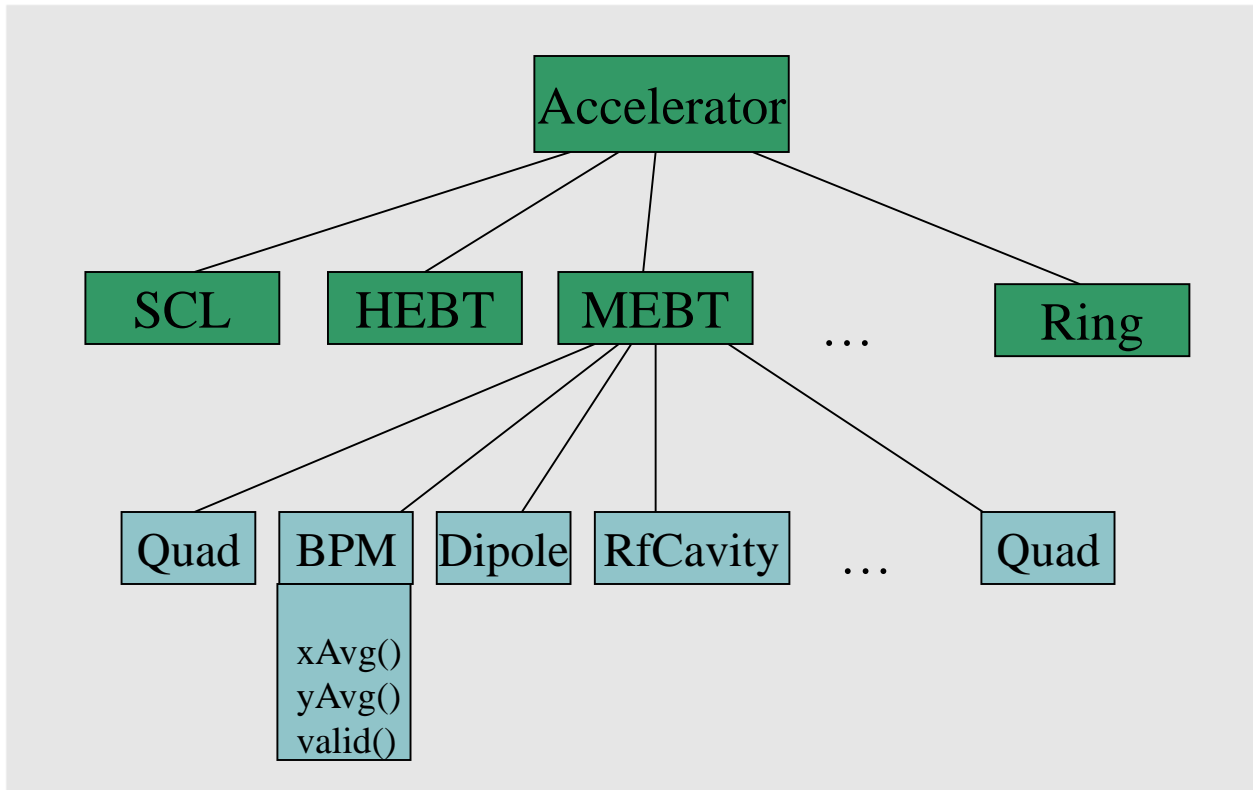
XAL includes:

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

XAL and its apps



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



**Initialized from the XML file**



**Created from  
Oracle Database**

- 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

<b>XAL Online Model</b>	A part of XAL/OpenXAL programming framework. Envelop and Single Particle Dynamics, inherited from Trace-3D and PARMILA
<b>PARMILA</b>	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.
<b>IMPACT</b>	It is a parallel computer PIC accelerator code which includes 3D space charge calculations. At SNS it was used for offline analysis.
<b>Trace-3D</b>	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)

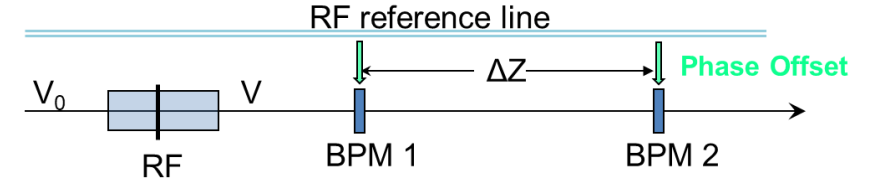
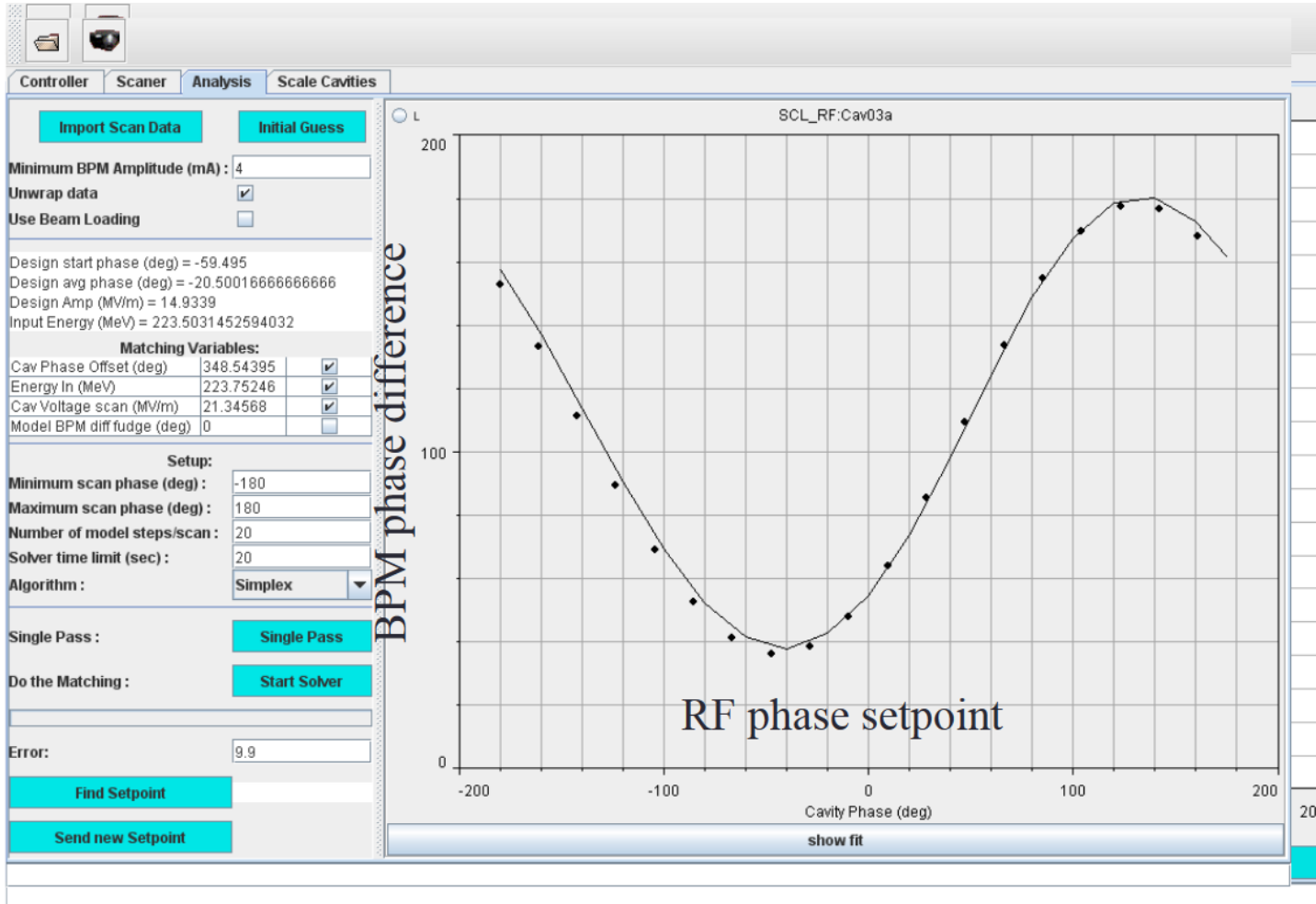
# List of Useful Apps for Commissioning

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)



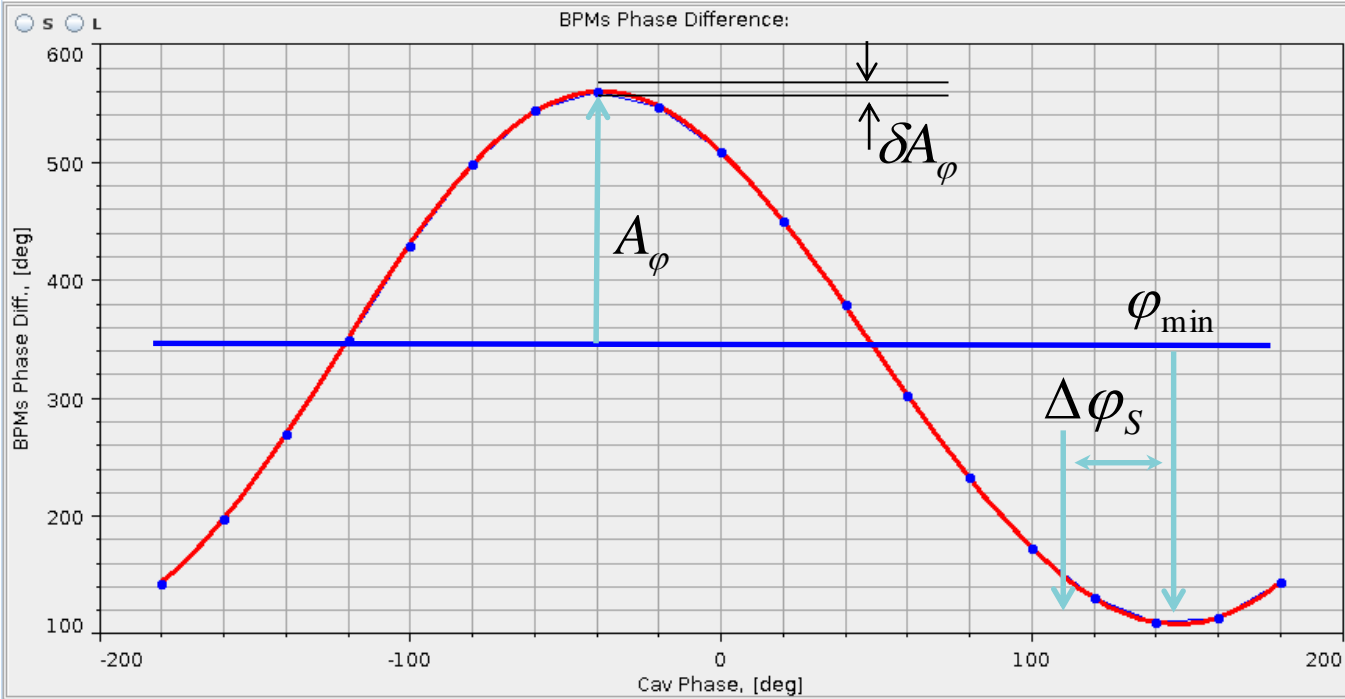
- Choose two BPMs
- All cavities between them are OFF
- Perform RF phase scan
- Find minimal phase diff. (max energy gain) and subtract  $180^\circ$
- Energy from TOF for two BPMs

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!

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

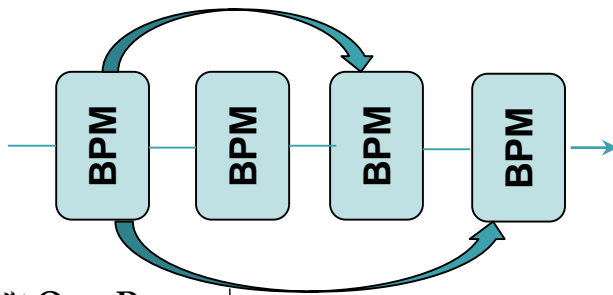
$$A_\varphi \approx \Delta z_{BPM} \cdot \frac{1}{(\gamma \cdot \beta)^3} \cdot E_0 TL$$

## 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!

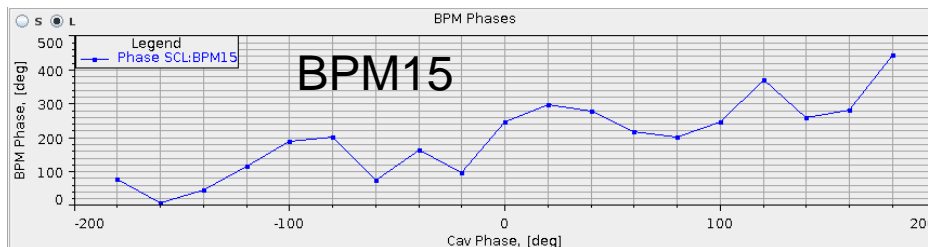
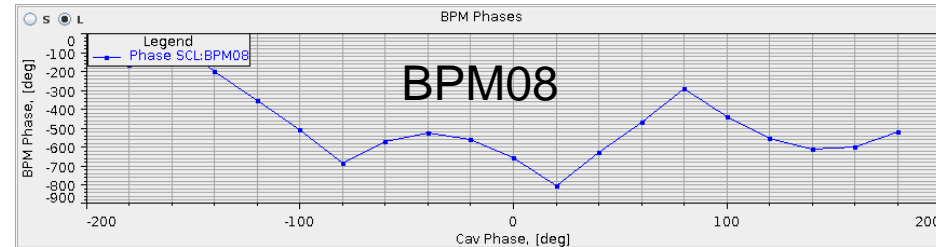
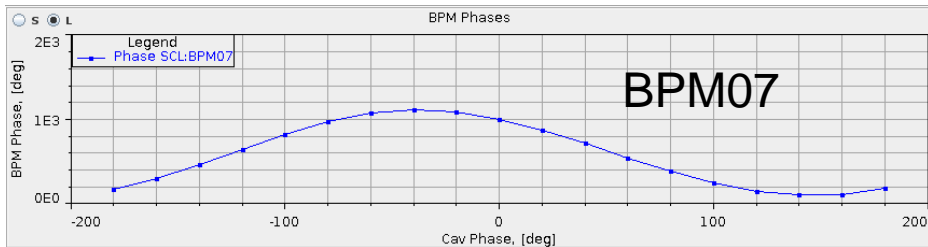
Less steps (N) – faster the scan!



# A “Big Phase Step” Problem

- BPMs measure phase in  $-180^{\circ}$  to  $+180^{\circ}$  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 than  $180^{\circ}$  gain in one step
- If we use far away BPM pairs, it could be problem for the “big phase step”

$$A_{\phi} \cdot \sin(\Delta\phi_{RF}) < 180^{\circ}$$



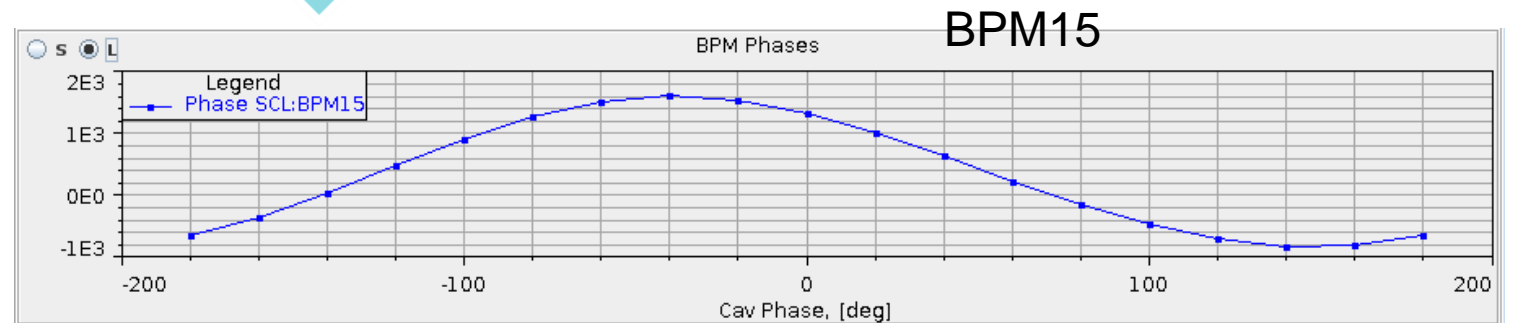
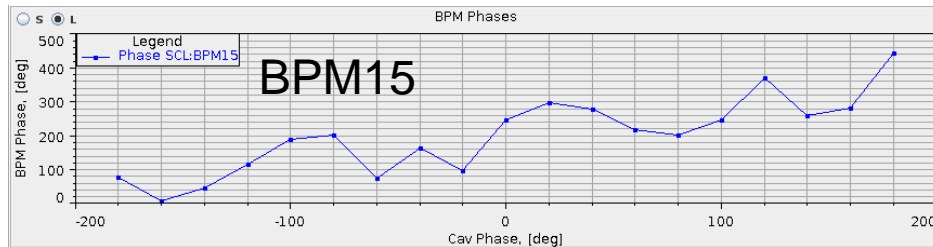
An example of Cav01a scan.  
We cannot go further BPM07 with the step size  $20^{\circ}$

# Solution for the “Big Step” Problem

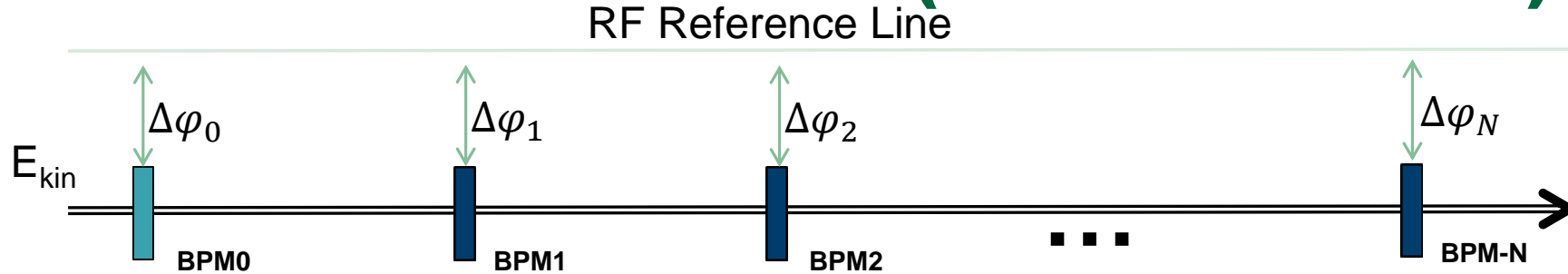
Most simple – iterative approach – the unwrapping is done by using not only the **previous point**, but also **the previous and current points from the previous BPM**. The iteration starts with the BPM closest to the cavity.

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

**Phase step size can be 40°, 60° or may be even 90°.**  
**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

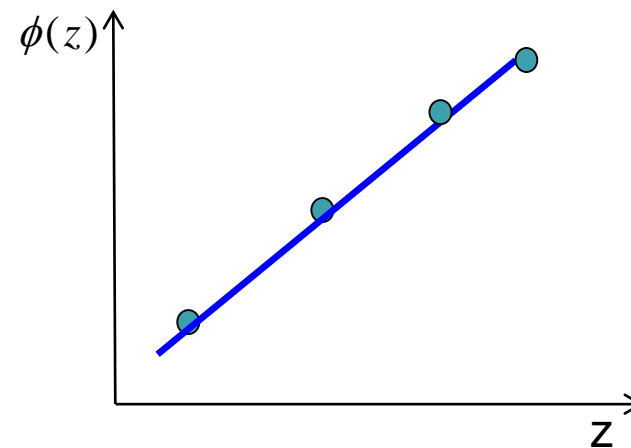
**After we know the BPM phase offsets, we can calculate the energy by reverting the formula**

# 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

We clearly see the calibrated pairs of BPMs.

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



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

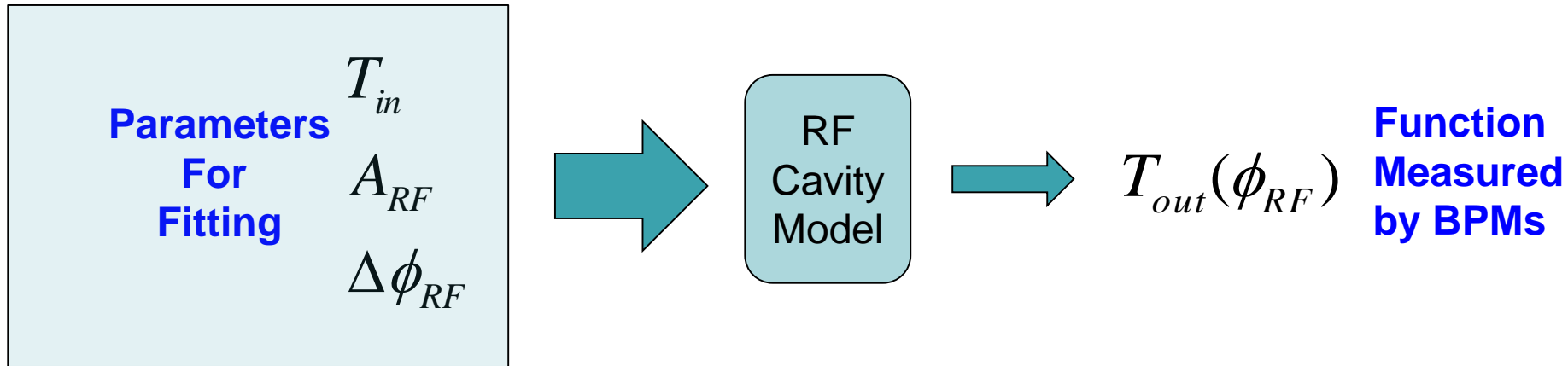
Now energy is found by using all available BPMs that are downstream of the scanned cavity.

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



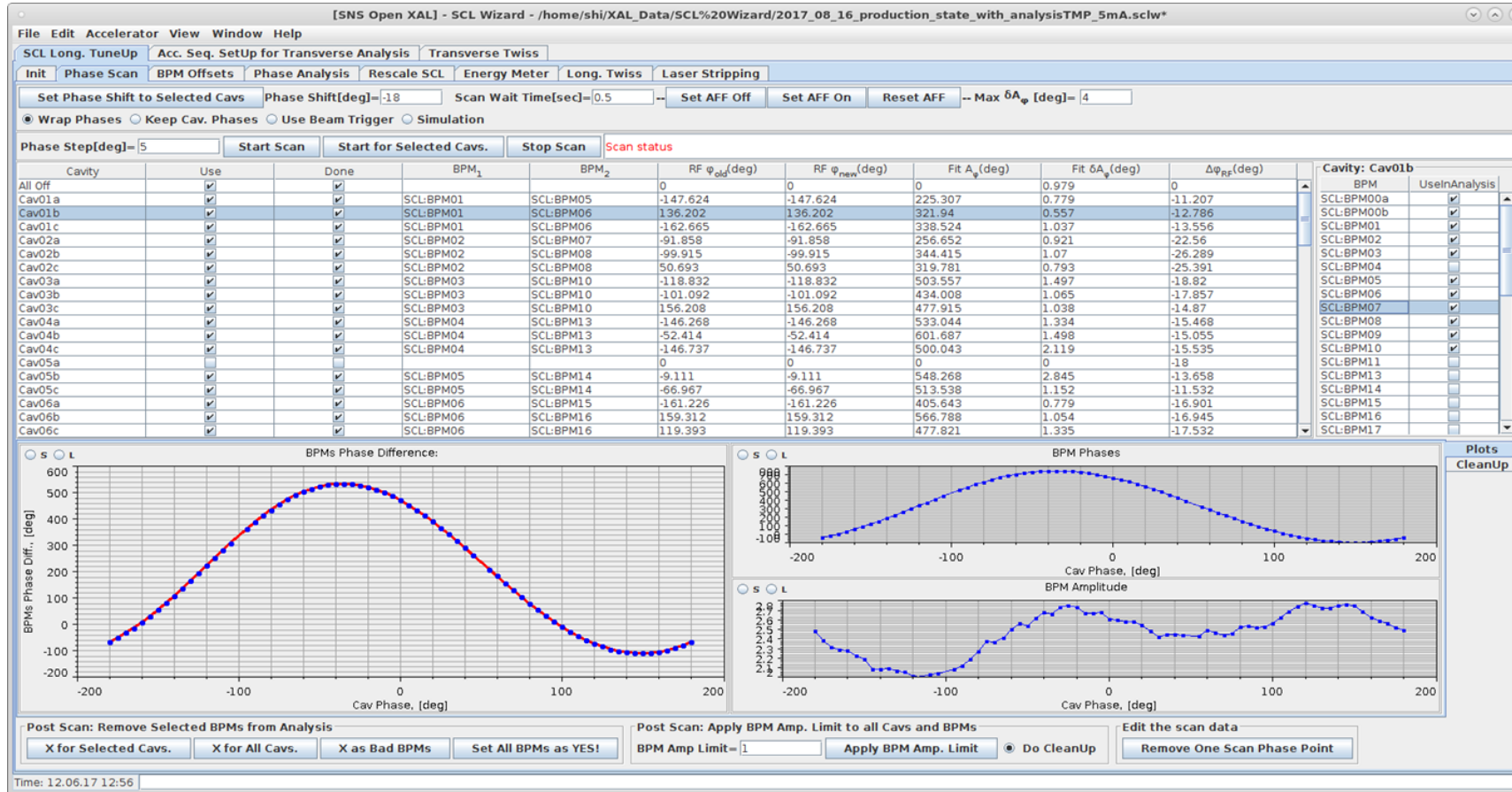
Accuracy of the energy calculation from two BPMs

# 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

[SNS Production XAL] - SCL Wizard - /home/shishlo/tmp/oooooo/SCL%20Wizard/2015\_11\_15\_scl\_production\_with\_LWs\_36mA.sclw\*

File Edit Accelerator View Window Help

Acc. Seq. Setup Transverse Twiss SCL Long. TuneUp

WS,LW Data Acquisition Transv. Twiss Analysis

Quad and Cavities Amp.&Phases Sets

# Use  Use Ext. Gauss Fit  Use Gauss Fit  Use RMS

Dump Quad Fields to ASCII  
 Dump. Cav Amps. Phases to ASCII  
 Read Cav Amps. Phases from ASCII

Horizontal

#	WS/LW	Pos.[m]	Use	S <sub>x</sub> [mm]	Gauss	RMS
1	SCL_Diag.LW01	7.37	<input checked="" type="checkbox"/>	3.72	3.82	4.12
3	SCL_Diag.LW02	13.21	<input checked="" type="checkbox"/>	2.54	2.67	2.1
5	SCL_Diag.LW03	19.05	<input checked="" type="checkbox"/>	1.71	1.71	3.15
7	SCL_Diag.LW04	24.89	<input checked="" type="checkbox"/>	3.37	3.12	3.66
9	SCL_Diag.LW12	73.65	<input checked="" type="checkbox"/>	2.36	2.3	3
11	SCL_Diag.LW13	81.54	<input checked="" type="checkbox"/>	2.58	2.52	3.91
13	SCL_Diag.LW14	89.43	<input checked="" type="checkbox"/>	3.46	3.52	4.3
15	SCL_Diag.LW15	97.32	<input checked="" type="checkbox"/>	3.15	3.1	3.54
17	SCL_Diag.LW32	231.47	<input checked="" type="checkbox"/>	2.41	2.44	2.51
19	SCL_Diag.LW01	7.37	<input checked="" type="checkbox"/>	3.59	3.7	4.11

Vertical

#	WS/LW	Pos.[m]	Use	S <sub>y</sub> [mm]	Gauss	RMS
2	SCL_Diag.LW01	7.37	<input checked="" type="checkbox"/>	2.57	2.57	2.97
4	SCL_Diag.LW02	13.21	<input checked="" type="checkbox"/>	2.48	2.51	2.35
6	SCL_Diag.LW03	19.05	<input checked="" type="checkbox"/>	1.7	1.63	1.81
8	SCL_Diag.LW04	24.89	<input checked="" type="checkbox"/>	1.65	1.61	2.32
10	SCL_Diag.LW12	73.65	<input checked="" type="checkbox"/>	3.16	3.11	3.04
12	SCL_Diag.LW13	81.54	<input checked="" type="checkbox"/>	3.62	3.71	2.93
14	SCL_Diag.LW14	89.43	<input checked="" type="checkbox"/>	3.89	4.07	3.19
16	SCL_Diag.LW15	97.32	<input checked="" type="checkbox"/>	3.3	3.68	2.8
18	SCL_Diag.LW32	231.47	<input checked="" type="checkbox"/>	0.74	0.73	1.2
20	SCL_Diag.LW01	7.37	<input checked="" type="checkbox"/>	2.56	2.68	3.05

Initial Twiss Parameters

Name	value	Error	Fit Step
Alpha X	-0.8468	0	0.1
Beta X	3.3467	0	0.1
Emitt X	1.1034	0	0.0167
Alpha Y	-0.3573	0	0.1
Beta Y	4.5872	0	0.1
Emitt Y	0.6259	0	0.0191
Alpha Z	0.2138	0	0
Beta Z	6.071	0	0
Emitt Z	0.3	0	0

eKin[MeV]=1.860400E2  
 Curr.[mA]=3.600E1  
 Fit Err.%=5.000

Fit Iterations=2E2  
 Iters. Left=0  
 Avg.Diff.[mm]=2.795533E-1

Make One Pass  
 Start Fitting  
 Stop Fitting

Init Twiss from Design

Final Twiss Fitting Results

Name	value	Error
Alpha X	-0.6479	0.0486
Beta X	3.3729	0.1996
Emitt X	1.0994	0.0476
Alpha Y	-0.3249	0.037
Beta Y	3.7734	0.2181
Emitt Y	0.5821	0.019
Alpha Z	0.2138	0
Beta Z	6.071	0
Emitt Z	0.3	0

Copy Results To Initial Twiss

Hor. Ver. Long. Plots Cavity and Quads Tables

Horizontal Size

Hor.

Vertical Size

Ver.

Longitudinal Size, RF Freq. = 402.5 MHz

Long.

# 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