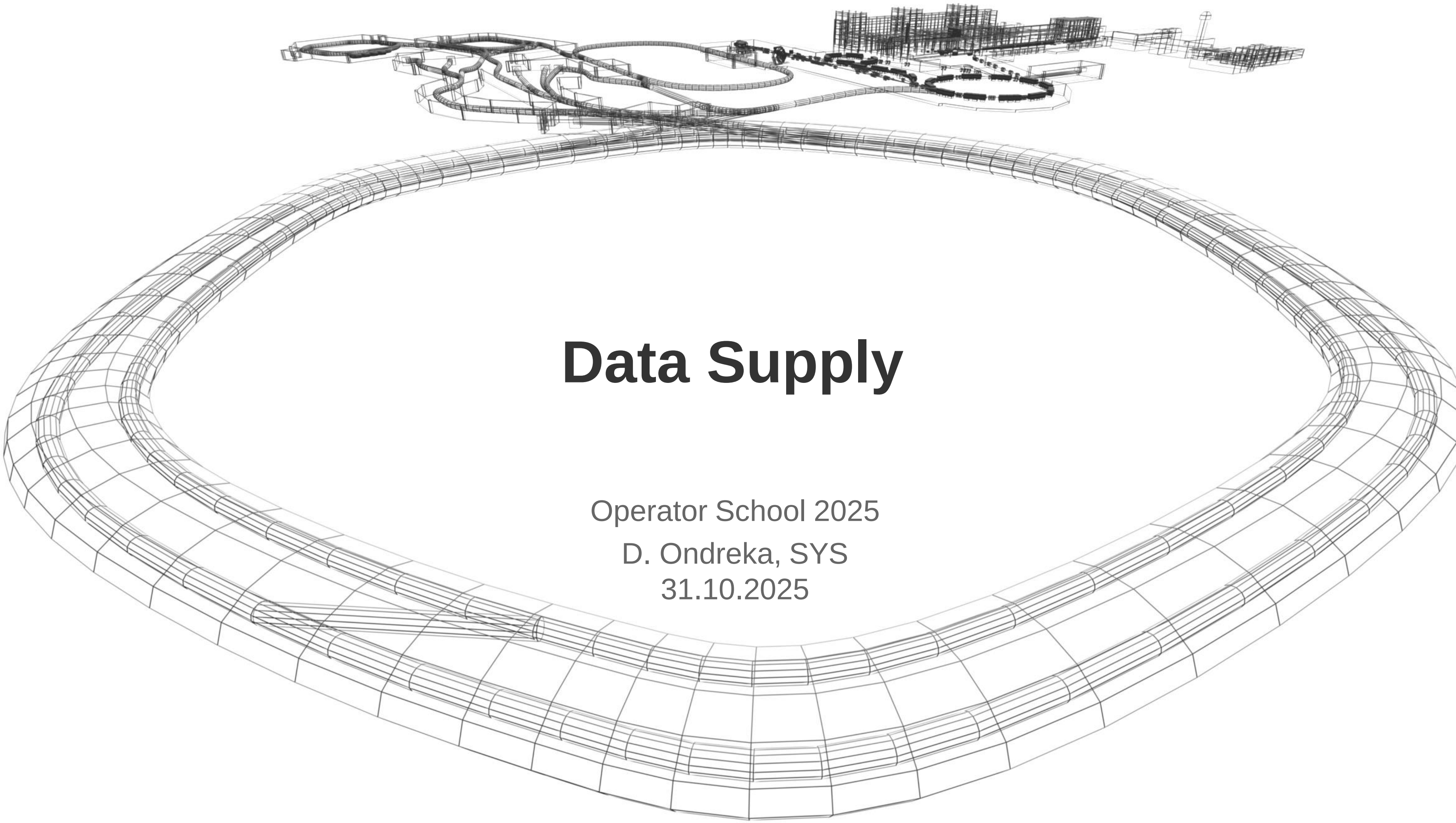


Data Supply

Operator School 2025

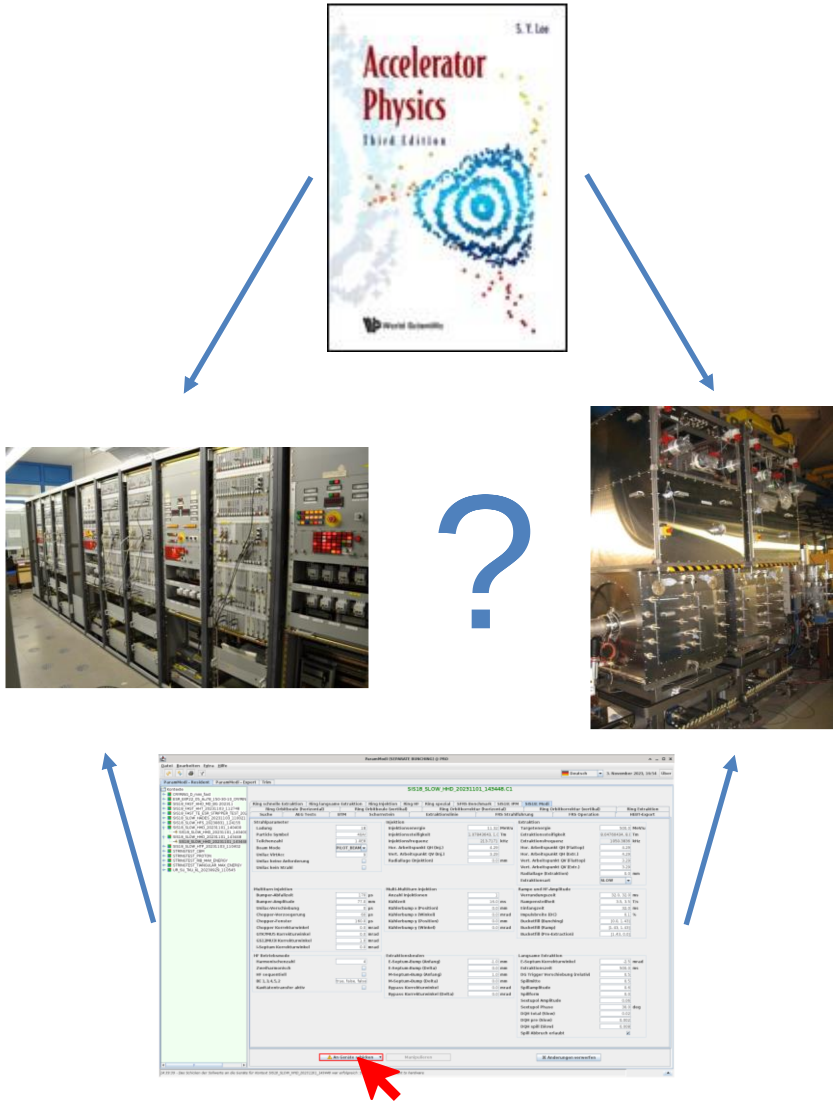
D. Ondreka, SYS

31.10.2025

Have you ever asked yourself…

- How does the control system know
  - how an accelerator is supposed to work?
  - what a synchrotron cycle is?
  - which knobs are needed to tune a machine?
  - the properties of all devices?

- If I change a parameter in ParamModi:
  - What happens behind the scenes?
  - How can I know which devices are affected?
  - What is a parameter anyway?
  - How can I inspect old and new values?



What is ‘Data Supply’?

- Accelerator can only work if
  - all devices have consistent set values
  - the timing system provides the right events
  - required beam parameters can be specified
  - required operation modes can be selected
  - the right knobs for tuning are available

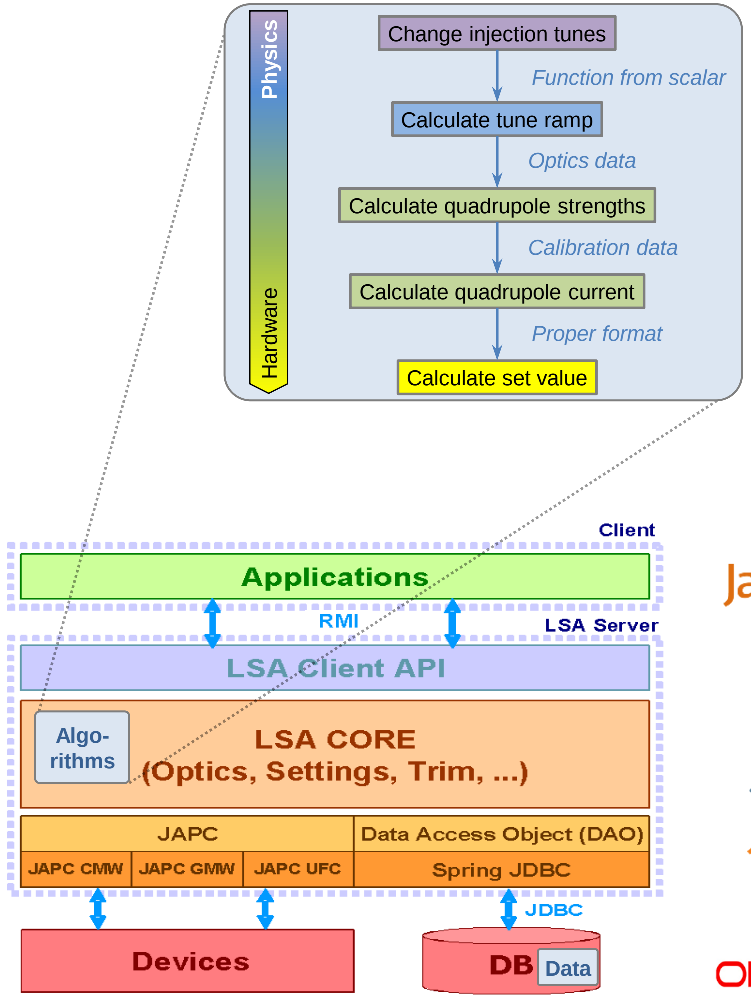
That’s what ‘Data Supply’ is about!

- Core feature: **machine model**
  - Abstraction in terms of physics parameters
  - Relations to hardware set values
  - Representation of operation modes

SIS18 Model Statistics	
#Devices	~150
#Parameters	~1500
#Input Parameters	~200
#Hardware Parameters	~350
#Rules	~1700

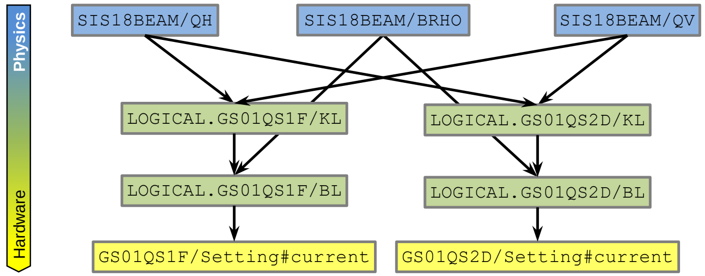
Machine modeling using LSA

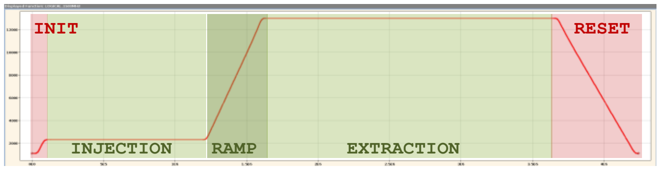
- LSA: Framework for accelerator modeling
  - Invented by CERN, co-developed by CERN/GSI
- Data-driven system with 3-tier architecture
  - LSA-DB: storage for input and output data
  - LSA-Server: calculations and data access
  - Applications: management of data
- Generic structures for modeling
  - Tables for machine specific data (e.g. optics)
  - Tables for device data (e.g. calibrations, limits)
  - Object model for description of accelerators
  - Interface for definition of physics algorithms



LSA in a nutshell

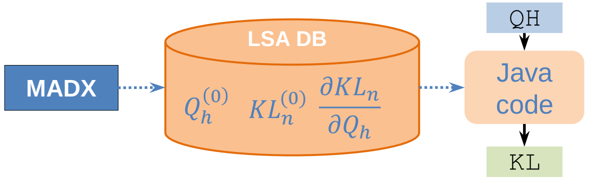
- Spatial partitioning of facility
  - **Particle transfers** (=timing groups)
  - **Accelerator zones**
- Accelerator represented via **parameters**
  - Linked to logical or physical devices
  - Organized in **hierarchy** from physics to hardware
- Cycles organized in contexts
  - Basic unit: **beam process** for particle transfer
  - Grouping into **chains** and sub-chains
- Values represented as **settings**
  - Value of parameter in beam process
  - HW settings get sent to devices (set-values)
- Calculations implemented as **rules**
  - Java code mostly written by physics experts
  - Calculate child from parents and data in LSA-DB





Setting of tune parameter	
Parameter	SIS18BEAM/QH
Beam process	P.SIS18.RING_INJECTION
Value	Constant function = 4.28

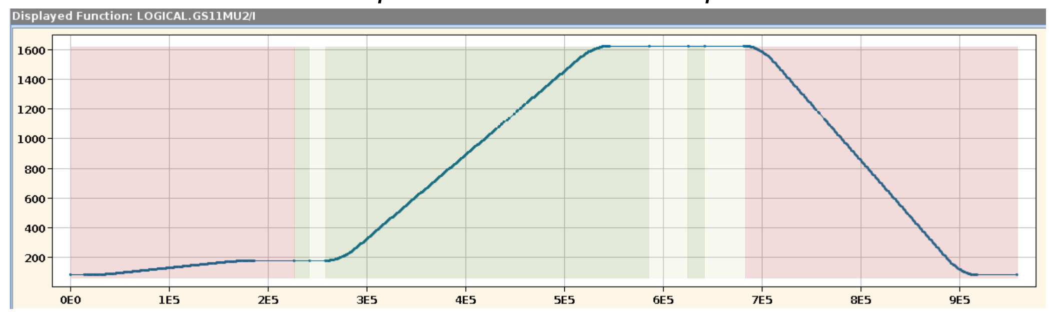
$$KL_n = KL_n^{(0)} + \frac{\partial KL_n}{\partial Q_h} \cdot (Q_h - Q_h^{(0)})$$

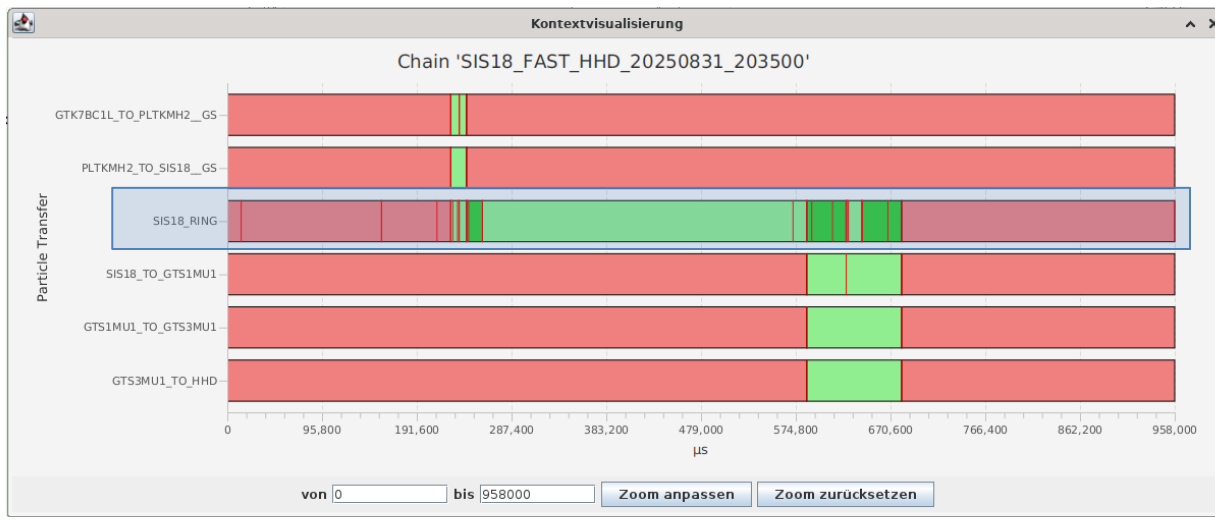


SIS18 Cycle

- Cycle represented as chain
  - Comprises several particle transfers
    - Ring, cooler, beamline sections
  - Partitioned into in beam processes
    - Types and number depend on particle transfer
  - Display with chain viewer
    - ParamModi: context menu on chain name
    - Displays timing events also

SIS18 dipole current and beam processes





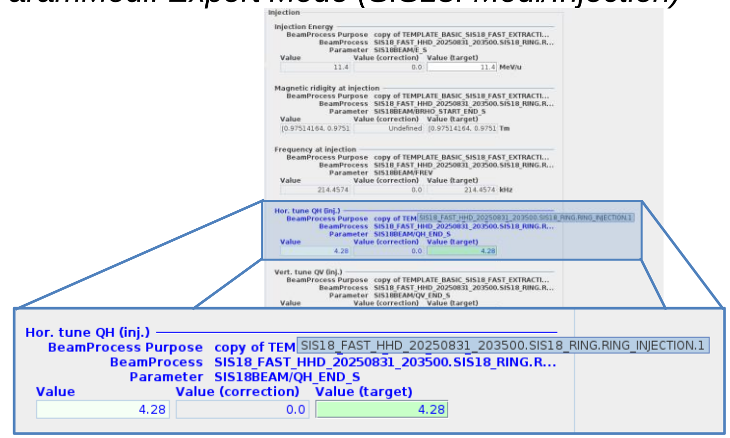
BP boundary
Timing event
BEAM_IN
BEAM_OUT

Hierarchy and Trim: An Example

- What happens if we change injection tune?
  - ParamModi sends changed value to LSA-server
  - LSA-server calculates new values along hierarchy
  - Let’s join it on the journey!

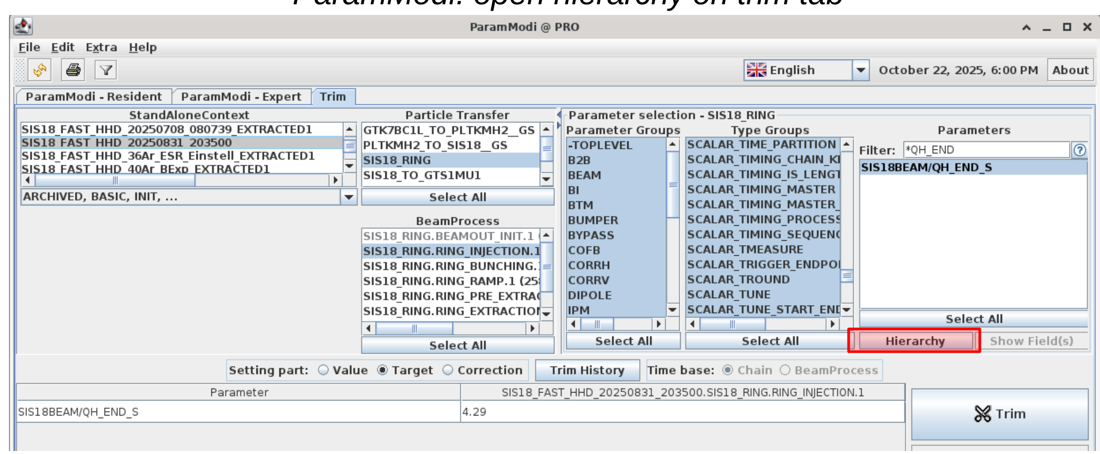
- How do I know the parameter?
  - Go to ‘Expert Mode’ (Menu ‘Extras’)
  - Find the input field by its label ‘Hor. tune QH (Inj.)’
    - Linked to parameter ‘SIS18BEAM/QH_END_S’
    - Linked to injection beam process

ParamModi: Expert Mode (SIS18: Modi/Injection)

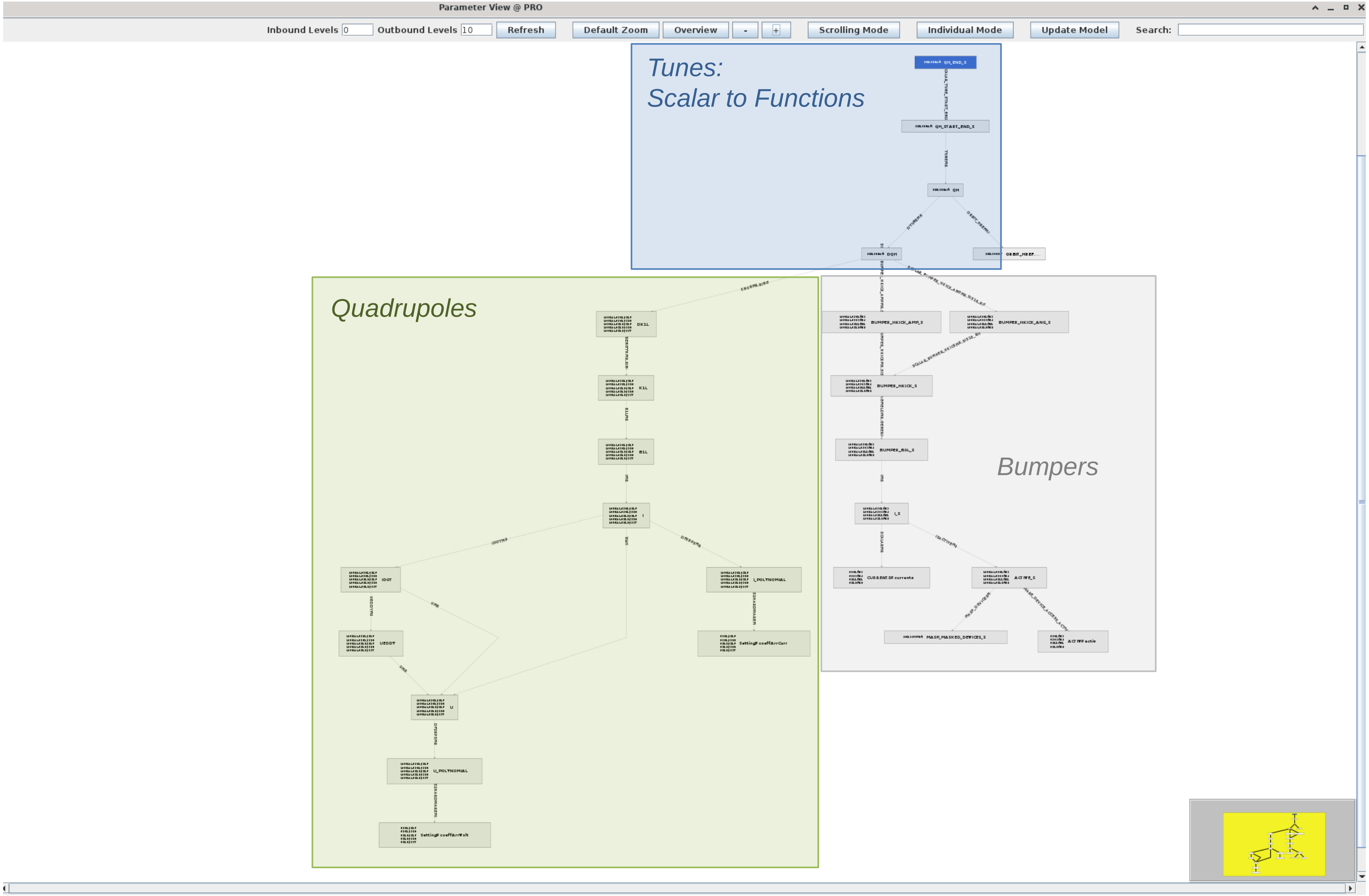


- Open hierarchy viewer on trim tab
  - Select parameter and click ‘Hierarchy’
  - How to find parameter?
    - Select all ‘Parameter Groups’
    - Select all ‘Type Groups’
    - Use ‘Filter’ field, e.g. ‘*QH_END’

ParamModi: open hierarchy on trim tab

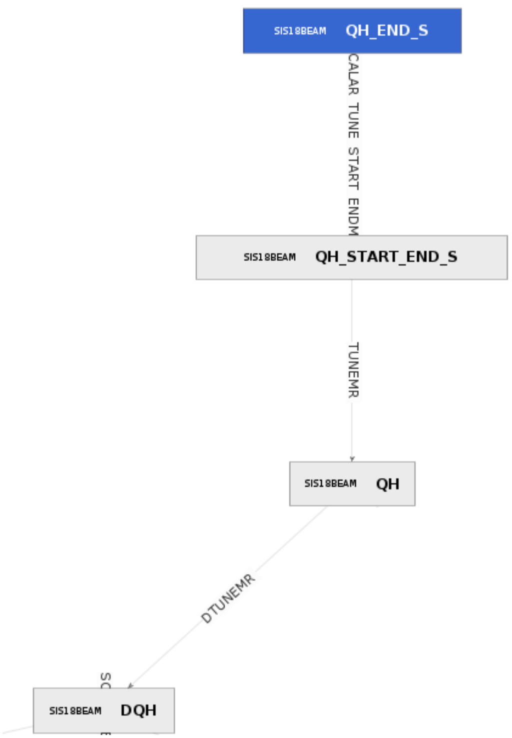


Tune Trim: Hierarchy



Tune Trim: Scalar to Functions

- Scalar input must be converted to function
  - Tune function required when there is beam
  - Tunes constant during ramp → change in PRE



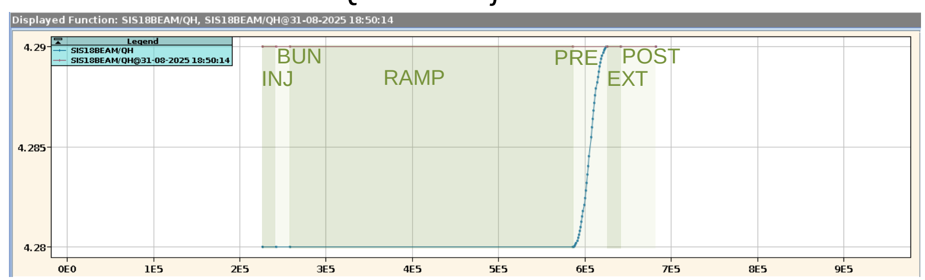
Input value for QH at injection

Parameter	SIS18_FAST_HHD_202508...	31-08-2025 18:50:14
SIS18BEAM/QH_END_S	4.28	4.29

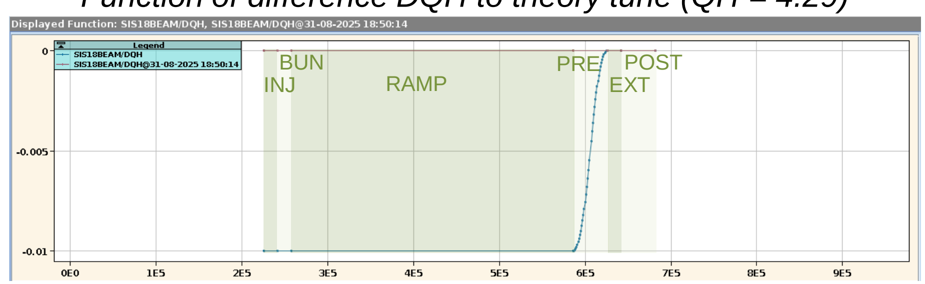
Values for QH at beam process boundaries

Parameter	SIS18_FA...	SIS18_FA...	SIS18_FA...	SIS18_FA...	SIS18_FA...	SIS18_FA...	SIS18_FA...	SIS18_FAS...
SIS18BEAM/QH_START_END_S		4.28, 4.28	4.28, 4.28	4.28, 4.28	4.28, 4.29	4.29, 4.29	4.29, 4.29	
		4.29, 4.29	4.29, 4.29	4.29, 4.29	4.29, 4.29	4.29, 4.29	4.29, 4.29	

Function for QH from injection to extraction



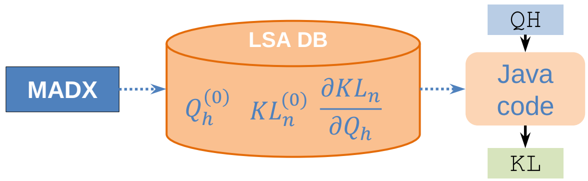
Function of difference DQH to theory tune (QH = 4.29)

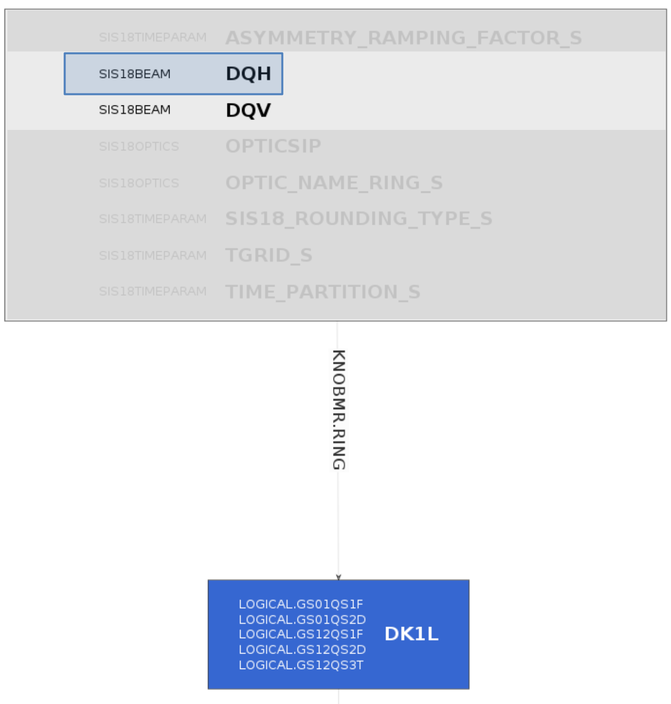


Tune Trim: Quadrupole Strengths

- Accelerator physics comes in:
  - Quadrupole strengths depend on both tunes
  - Generally requires ion optical simulations
  - Linear approximation used in model

$$KL_n = KL_n^{(0)} + \frac{\partial KL_n}{\partial Q_h} \cdot (Q_h - Q_h^{(0)})$$

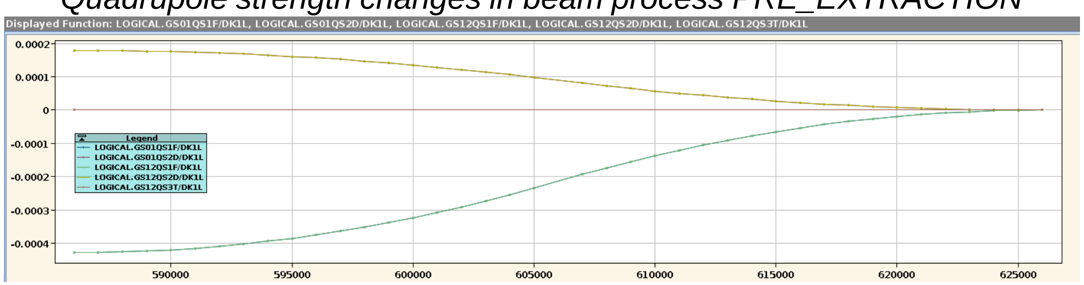




Coefficients for linear approximation of strength change

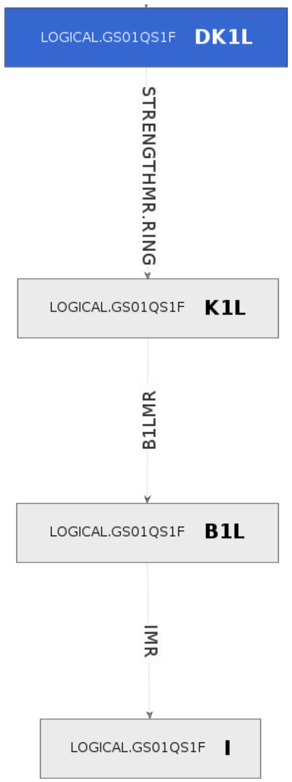
```
KNOBS;SIS18_TAU_1.000
KNOB NAME      ; COMPONENT NAME          ;  KNOB VALUE
SIS18BEAM/DQH ; LOGICAL.GS01QS1F/DK1L ;  0.0427877463927
SIS18BEAM/DQH ; LOGICAL.GS12QS1F/DK1L ;  0.0427877463927
SIS18BEAM/DQH ; LOGICAL.GS01QS2D/DK1L ; -0.0179062106345
SIS18BEAM/DQH ; LOGICAL.GS12QS2D/DK1L ; -0.0179062106345
SIS18BEAM/DQH ; LOGICAL.GS12QS3T/DK1L ;  0.0
SIS18BEAM/DQV ; LOGICAL.GS01QS1F/DK1L ;  0.0231789186936
SIS18BEAM/DQV ; LOGICAL.GS12QS1F/DK1L ;  0.0231789186936
SIS18BEAM/DQV ; LOGICAL.GS01QS2D/DK1L ; -0.0567714510034
SIS18BEAM/DQV ; LOGICAL.GS12QS2D/DK1L ; -0.0567714510034
SIS18BEAM/DQV ; LOGICAL.GS12QS3T/DK1L ;  0.0
```

Quadrupole strength changes in beam process PRE_EXTRACTION

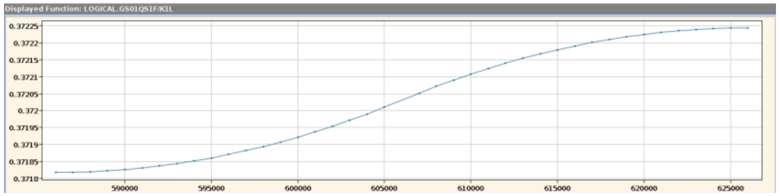


Tune Trim: Strength to Current

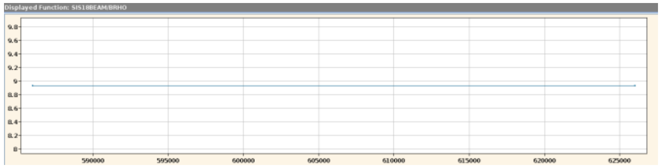
- Independent calculations per quadrupole
  - Integral field: strength x rigidity
  - Current: calibration curve look-up



Strength (K1L): DK1L + theory strength

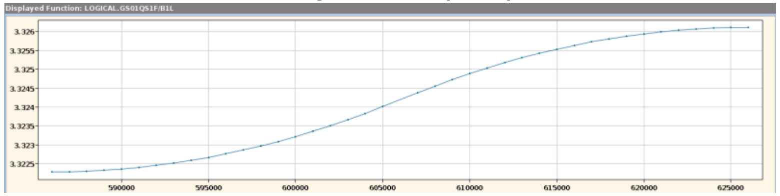


Rigidity (BRHO)

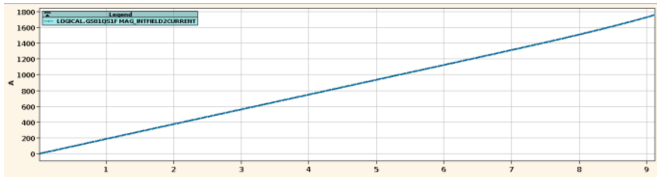


x BRHO

Integral field (B1L)

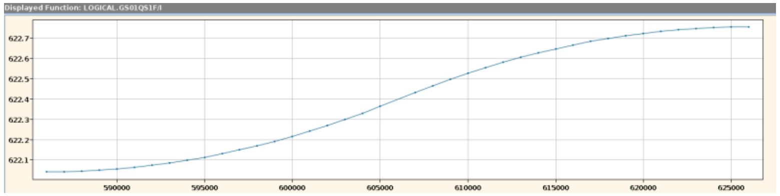


Calibration curve



Calibration look-up

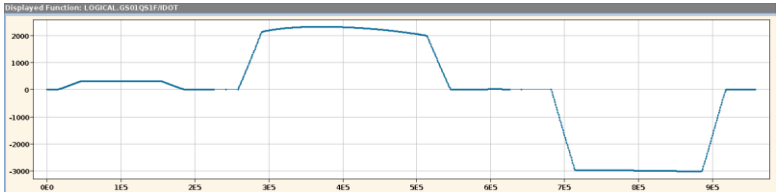
Quadrupole current (I)



Tune Trim: Current to Voltage

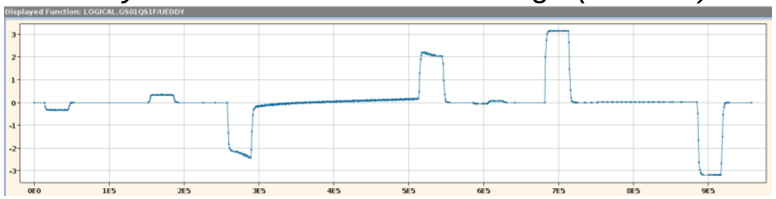
- Power converters need current and voltage
  - Inductive contribution from current change
  - Eddy current contribution

Current change (IDOT): numerical differentiation

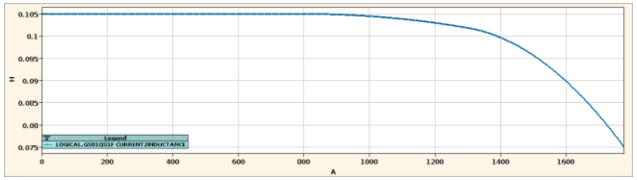


Eddy current model

Eddy current contribution to voltage (UEDDY)

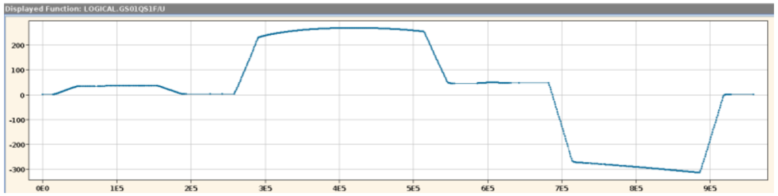


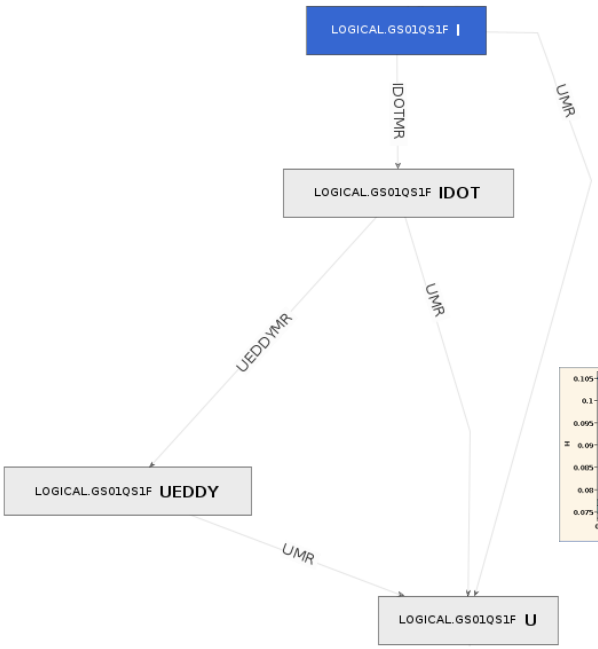
Inductance curve



$U = R \cdot I + L(I) \cdot dI/dt + U_{eddy}$

Load voltage prediction (U)

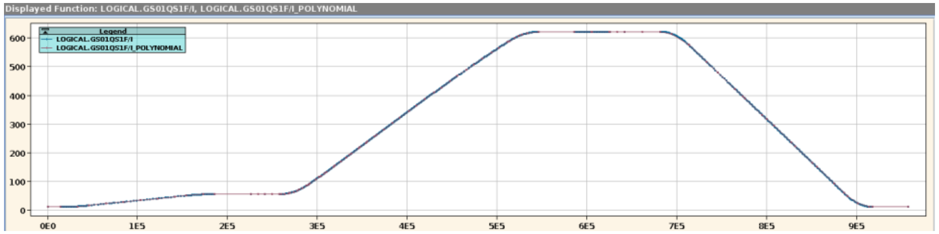




Tune Trim: Set Values

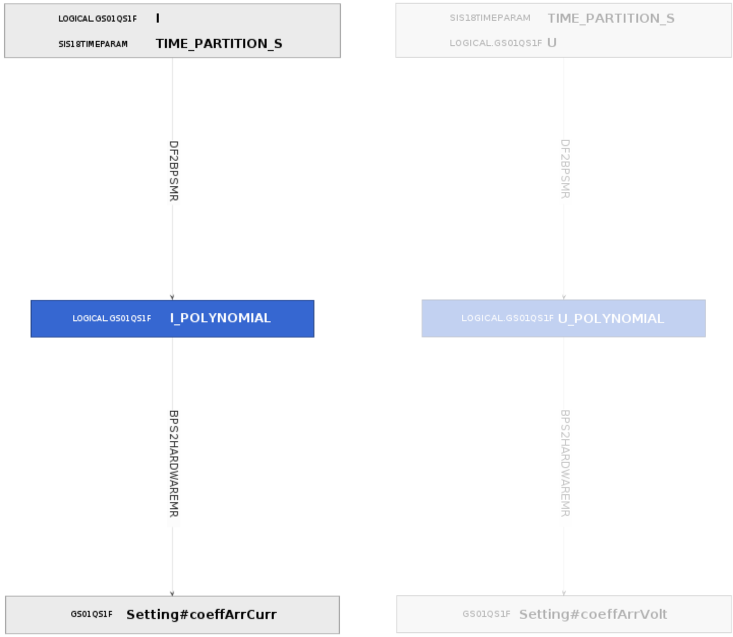
- Ramped SCUs need quadratic polynomials
  - Polynomial fit of values on time grid
  - Extraction of coefficients into set value parameters
  - Sent to SCU for resident chains

Current polynomial (I_POLYNOMIAL) and current (I)



Current set value Setting#coeffArrCurr (RAMP)

	0	1	2	3	4	5
0	0.0	0.002	54.78547	0.18000000...	31442.4999...	0.0
1	0.002	0.006	54.9116	125.812499...	31648.7500...	0.0
2	0.006	0.01	55.92123	378.872500...	32091.2499...	0.0
3	0.01	0.012	57.95018	636.434999...	32332.5000...	0.0
4	0.012	0.016	59.35238	764.612500...	32733.7499...	0.0
5	0.016	0.02	62.93457	1029.75999...	32925.0000...	0.0
6	0.02	0.022	67.58041	1297.13249...	34183.7500...	0.0
7	0.022	0.026	70.31141	1430.14000...	35319.9999...	0.0
8	0.026	0.03	76.5970899...	1710.66500...	35682.4999...	0.0
9	0.03	0.032	84.01067	1996.81249...	36713.7500...	0.0
10	0.032	0.034	88.15115	2144.97749...	2431.25000...	0.0
11	0.034	0.04	92.4508299...	2154.42458...	2850.62499...	0.0
12	0.04	0.044	105.479999...	2188.91749...	2373.75000...	0.0
13	0.044	0.046	114.273649...	2206.79374...	2170.62499...	0.0
14	0.046	0.05	118.69592	2216.11500...	2204.99999...	0.0
15	0.05	0.052	127.595659...	2234.14249...	1843.75000...	0.0
16	0.052	0.06	132.071320...	2240.53152...	1764.34027...	0.0
17	0.06	0.062	150.108490...	2268.84000...	1292.49999...	0.0
18	0.062	0.068	154.65134	2272.72416...	1433.75000...	0.0
19	0.068	0.074	168.339299...	2290.50958...	910.624999...	0.0
20	0.074	0.078	182.115139...	2301.85750...	1071.24999...	0.0
21	0.078	0.08	191.33971	2310.80375...	648.124999...	0.0
22	0.08	0.088	195.96391	2311.98888...	800.138888...	0.0
23	0.088	0.09	214.51103	2324.02999...	360.000000...	0.0
24	0.09	0.096	219.160529...	2324.89583...	563.749999...	0.0
25	0.096	0.098	233.1302	2331.96874...	195.624999...	0.0
26	0.098	0.106	237.794919...	2331.48652...	316.215277...	0.0
27	0.106	0.108	256.46705	2336.36499...	-44.999999...	0.0
28	0.108	0.116	261.139599...	2335.32500...	124.374999...	0.0
29	0.116	0.126	279.83016	2335.78491...	-102.29166...	0.0
30	0.126	0.134	303.177780...	2331.86138...	-206.73611...	0.0
31	0.134	0.144	321.819439...	2327.08991...	-409.79166...	0.0
32	0.144	0.152	345.049359...	2316.95472...	-507.15277...	0.0
33	0.152	0.162	363.552539...	2307.66220...	-705.52083...	0.0
34	0.162	0.172	386.558610...	2292.27062...	-841.56250...	0.0
35	0.172	0.18	409.39716	2273.85138...	-925.48611...	0.0
36	0.18	0.19	427.528739...	2257.54264...	-1097.8645...	0.0
37	0.19	0.198	449.99438	2233.83791...	-1188.6458...	0.0
38	0.198	0.208	467.78901	2213.46452...	-1360.0520...	0.0
39	0.208	0.216	489.78765	2184.24138...	-1448.6111...	0.0
40	0.216	0.226	507.16887	2159.91329...	-1658.2291...	0.0
41	0.226	0.236	528.602179...	2125.41041...	-1801.0416...	0.0
42	0.236	0.244	549.67618	2087.52250...	-1896.2500...	0.0
43	0.244	0.254	566.255	2055.48097...	-2108.6979...	0.0
44	0.254	0.256	586.598940...	2012.56125...	-2945.6249...	0.0
45	0.256	0.262	590.612280...	2001.35333...	-32944.999...	0.0
46	0.262	0.266	601.434379...	1604.17250...	-31666.250...	0.0
47	0.266	0.27	607.344409...	1346.34999...	-30927.500...	0.0
48	0.27	0.276	612.23497	1094.77791...	-30721.875...	0.0
49	0.276	0.288	617.697650...	723.775000...	-30157.291...	0.0
50	0.288	0.328	622.0403	0.0	0.0	0.0



Model: Interface to Machine

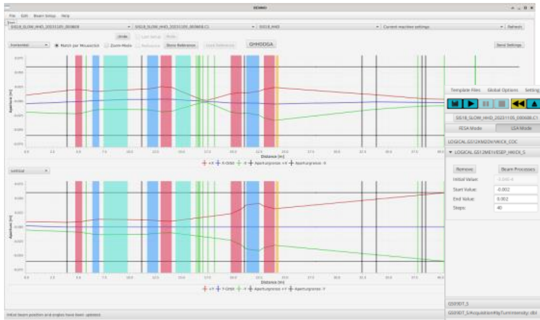
- Model provides general access to settings
  - Modifications not limited to ParamModi
  - Programmatic access to all levels of the hierarchy
  - Modification of time functions possible

Java code for executing the tune trim just discussed

```
final StandAloneContext chain = Services.getContextService().findStandAloneChain("SIS18_FAST_HHD_20250831_203500");
final Parameter qhParameter = Services.getParameterService().findParameterByName("SIS18BEAM/QH_END_S");
final BeamProcess injBeamProcess = Contexts.getBeamProcessByPurpose(
        chain.getBeamProcesses(), GsiBeamProcessPurpose.RING_INJECTION);
final ImmutableValue qhInj = ValueFactory.createScalar(4.28);
final TrimRequestBuilder trimBuilder = new TrimRequestBuilder();
trimBuilder.setContext(chain);
trimBuilder.setSettingPart(SettingPartEnum.TARGET);
trimBuilder.setDescription("Execute my trim.");
trimBuilder.addValue(qhParameter, injBeamProcess, qhInj);
Services.getTrimService().trimSettings(trimBuilder.build());
```

- Applications using model in routine operation
  - Benno Application *(C. Heßler/OPE)*: beamline steering and focusing
  - Device Automator *(W. Geithner/DEC)*: parameter scanning

Benno Application

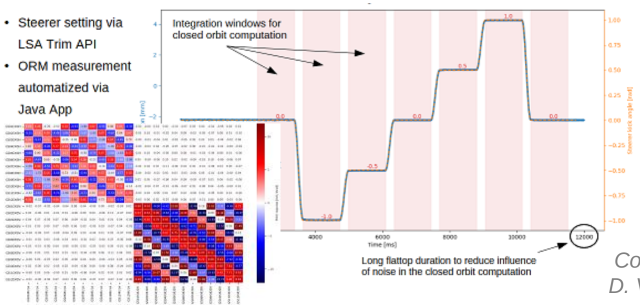


Device Automator Application

- Interfaces used for machine experiments
  - Java API: standard used by CS applications
  - Python bridge: experimental *(→ S. Appel/APH)*

LOCO-based ORM measurement

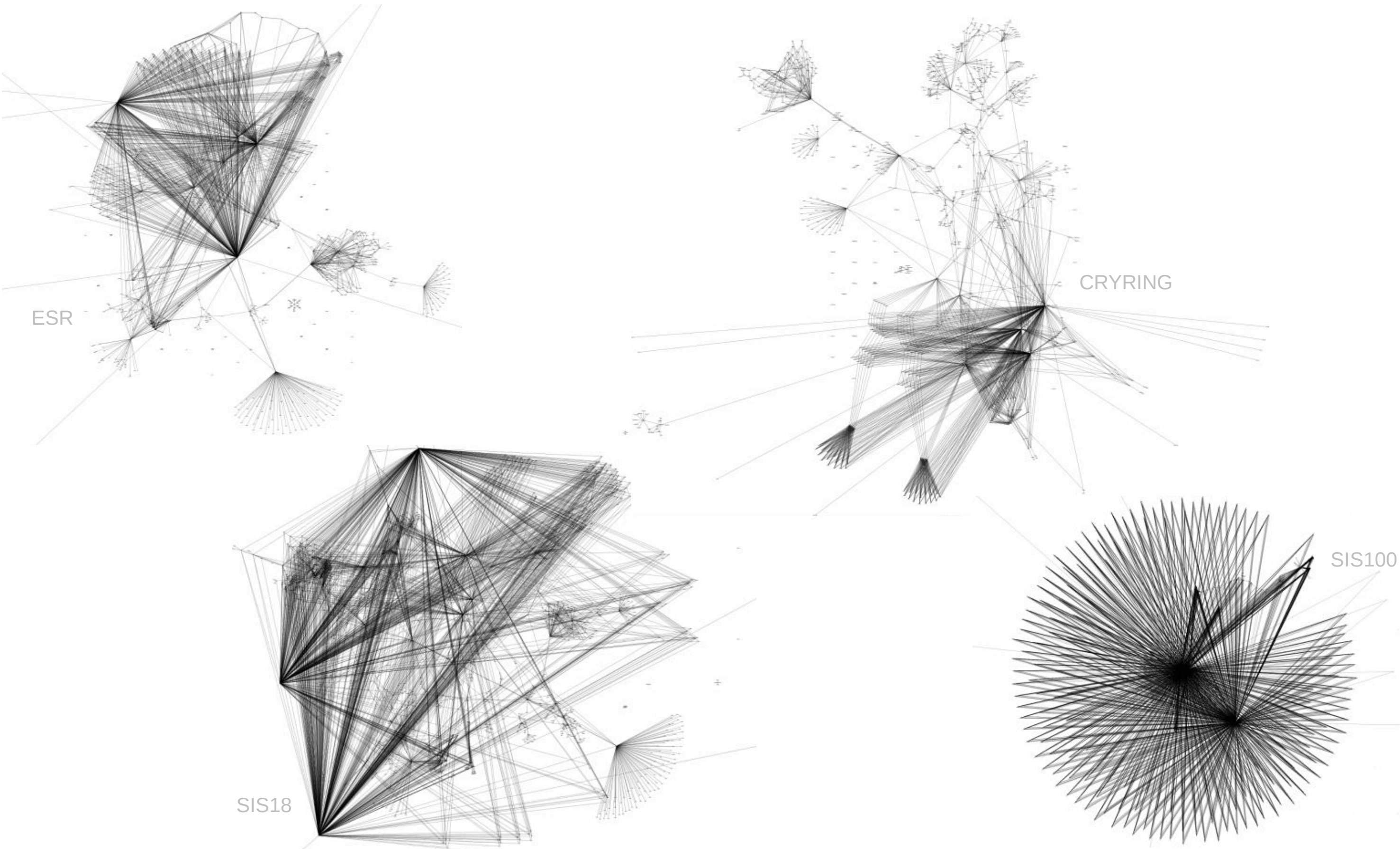
*(**Java app**, steerer function trims)*



Courtesy of D. Vilsmeier

Thank you for your attention!

Special thanks go to the great FAIR Data Supply team: Peter Gerhard, Holger Liebermann, Ingrid Kraus, Sergey Litvinov, Michael Kelnhofer, Christoph Hessler, Raphael Müller, Andreas Schaller, Jonas Pforr, Jutta Fitzek, Hanno Hüther, and others



Looks like accelerator modeling is an art…

Courtesy A. Schaller