Operation with Middle Layer:
Focus on Its Benefits

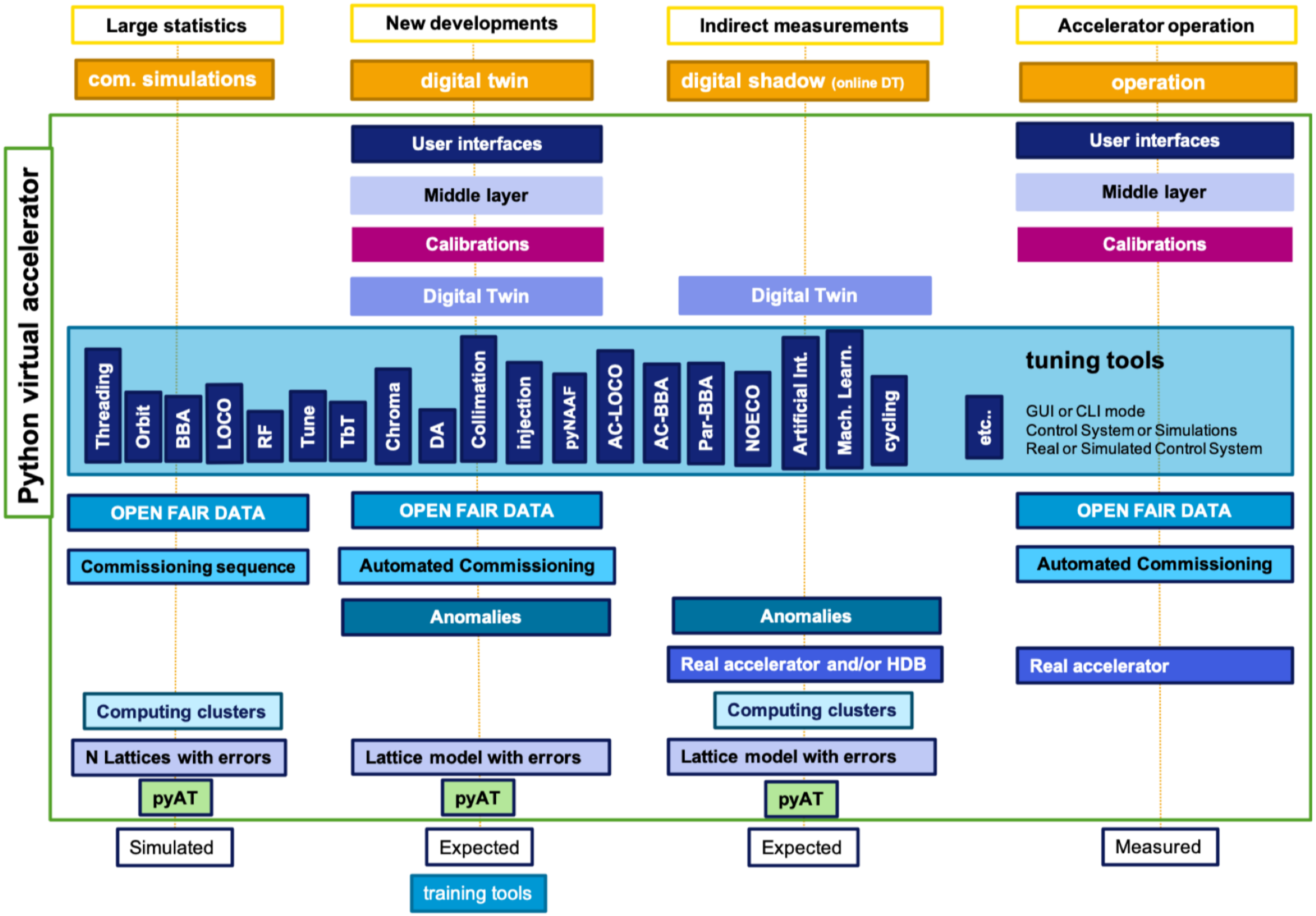
Laurent S. Nadolski
Accelerator Physics Group
Accelerators and Engineering Division
Synchrotron SOLEIL

Wednesday 19 June 2024

Presentation Layout

- Middle Layer Introduction
- What made the success of the Matlab Middle Layer
- Focus Relevant Features
- Questions

Warning: What this talk will not cover



No fancy Application Packages

No high-level applications

Focus on the definition of a Middle Layer

Starting from the G. Portmann's Conclusions: MML Success

See details in Gregory Portmann's MML History talk (2024), Laurent Nadolski's MML Status talk (2023)

- Relatively easy to use. Most people start writing useful scripts in a few hours (short learning curve).
- Middle Layer + LOCO + AT cover many high-level software concerns for storage ring physics. Hence, no need to spend resources coding the same algorithms.
- Thousands of dedicated accelerator hours have been spent testing, improving, debugging, and exercising the Middle Layer software. Reliability and Robustness
- It's a good scripting language for machine shifts or it can be the high-level setup and control software for a storage ring.
- Integration of the AT model is good for debugging software without using accelerator time.
- Having machine-independence software has fostered collaboration and code sharing between the laboratories.

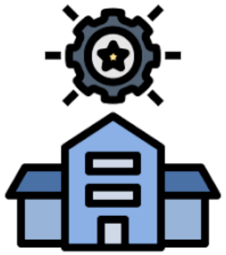
Middle Layer Keys for such a Success

- Easy to adopt (no need to be an expert in controls, in AT, etc.) → Minimum interface
- Short learning curve
- Easy to manipulate (families, function aliases…)
- Easy to maintain and update
- Well documented with examples
- Make use of generic functions
- Control Agnostic

- **Middle Layer configuration**

Baseline used by all low and high-level applications

ML config



Facility Choice (SOLEIL, ALS, etc.) Accelerator Choice (Ring, Booster, transfer lines…)

Accelerator Model Config

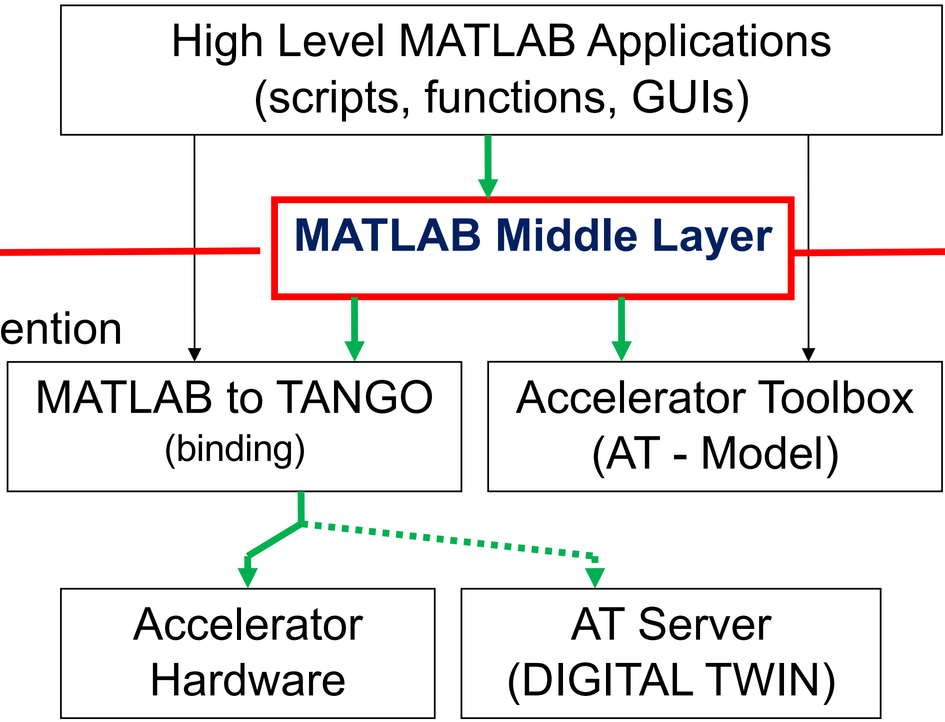
Data Storage

- Ready to be used
  - In the office
  - Using a laptop
  - In a control room
  - In connection with a digital Twin
  - In development platform
- Shortens the commissioning, and development time
- Generic tools for Beam-Based measurements (Machine development days, [early operation])

Why a Middle Layer?

Replace TANGO by your own control system

On principle AT could be replaced by another accelerator code



Hidden to users

- TANGO system
- TANGO naming convention

Device name:

ANS-C01/DG/BPM.2

Attribute name:

XPosSA (horizontal position/slow acquisition @ 10 Hz)

Hidden to users

- Low-level AT routines
- use of preformatted functions with default arguments

Equipment name (Physicist Natural Language):
'BPMx' Family

Signal name:
x, Px, s-position, Twiss parameters

Main Key Elements to Setup the Middle Layer

ML Configuration	Data Storing Configuration	ML / Model
Control Configuration (TANGO, EPICS, others)	Model Configuration (AT, others)	ML / Operation (controls)
Structured Data Self Descriptive	Interfaces to Interaction with Families	ML / High level control apps

Equipment Families
- MML features
- Model features
- Control features

Self Documented, with examples

Middle Layer Centric, fully coherent

Knowledge Database

Metadata (context, configuration, accelerator status)

A shared Accelerator Structure accessible by all applications

Once configured, the "magic" ML happens

- The ML provides the users and all the functions with all the information to work seamlessly
- Library of accelerator-oriented functions, GUIs, plotting, data analysis, and scripts are made generic
- No need to modify other ML functions

ML Basic Configuration

- Once the ML layer is configured, configuration data are made available in all applications
  - Use of setappdata or getappadata
  - Extensive use of aliases
  - Full directory path never specified literally in a script, function
  - Facility specific Information
- Paths to control, data storage
- Accelerator Specific Settings

Command	Alias	Output
`getfamilydata('Machine')`	getmachinename	SOLEIL
getfamilydata('SubMachine');	getsubmachinename	Storage Ring
getfamilydata('Directory', 'OpsData');		$MLROOT/dataroot4mml/machine/SOLEIL/StorageRingOpsData/superbend_rock_run4_2021
getfamilydata('Directory')		

Family Configuration

- Family: Equipment to interact with
  - Real set of equipment: BPM, Power Supplies, DCCT (current measurement)
  - Logical equipment, computation result: (Lifetime)

Field name	Value	
FamilyName	BPMx	
DeviceList	[Cell# Element#]	[122×2 double]
Status	1 or 0	Global, Local
Monitor	ANS-C14/DG/BPM.1/XPosSA	
Golden	12 µm	[122×1 double]
AT	ATIndex, s-position etc.	[122×1 double]
FamilyType	BPM	
MemberOf	'PlotFamily' 'Archivable'	
Gain, Roll, Crunch	Calibration	

```
           DataType: 'Scalar'
    HW2PhysicsParams: [122×1 double]
             HWUnits: 'mm'
             Handles: [122×1 double]
                Mode: 'Simulator'
    Physics2HWParams: [122×1 double]
        PhysicsUnits: 'm'
  SpecialFunctionGet: 'gethbpmgroup'
          TangoNames: {122×1 cell}
               Units: 'Hardware'
```

showfamily

Bending Magnet – BEND
Quadrupoles – Q1, Q2, … Q11
Sextupoles – S1, S2, … S12
Skew quadrupoles – QT
Correctors – HCOR, VCOR
Beam Position Monitors – BPMx and BPMy
Others - RF, DCCT, TUNE, GeV

Fields (any control data can be mapped)

Setpoint, Monitor, RampRate, TimeConstant, Sum, RunFlag,
Trim, FF, DAC, On, Reset, Ready, Voltage, Power, Velocity, UserGap, HallProbe, Limit, etc…

Naming Conventions (Nomenclature)

Power supply: 4th corrector in cell 2 of the storage ring

- Easy Access to Elements like in the model
  - DeviceList [Cell#, Element#]
  - ElementList

getam('HCOR, [2 4]')

11th correctors of the ring
ANS-C02/AE/S10.1-CH/current

- Ease communication (controls group, Accelerator Support Group)

- Maintenance/Servicing

Functions to facilitate correspondence between naming conventions

- family2dev
- family2elem
- family2tango
- family2tangoname / family2channelname
- family2common

Interaction Interface with Families

- Two (three) main very generic functions
  - Reading value (getpv)
  - Setting value (setpv)
  - Incremental value (steppv)
  - Command (TANGO specific)

- Derivative functions, aliases to ease the use
  - getsp (setpoint)
  - getam (monitor values)
  - getx (BPM position)
  - getrf (RF frequency reading)
  - getdcct (stored beam current)
  - ….

Error handling, setting tolerances
Min/Max check (safe operation)
Hysteresis of magnets

- Rich Interface
  - Polymorphism
  - Recursive Mechanism

```
% FamilyName/DeviceList Method
% [AM, tout, DataTime] = getpv(Family, Field, DeviceList, t, FreshDataFlag, TimeOutPeriod)
%
% Data Structure
% [AM, tout, DataTime] = getpv(DataStructure, t, FreshDataFlag, TimeOutPeriod)
%
% TangoName or TangoName Method
% [AM, tout, DataTime, ErrorFlag] = getpv(TangoName, t, FreshDataFlag, TimeOutPeriod)
%
% CommonName Method (Family can be '' to search all families, Field is optional)
% [AM, tout, DataTime] = getpv(CommonName, Field, t, FreshDataFlag, TimeOutPeriod)
% [AM, tout, DataTime] = getpv(Family, Field, CommonName, t, FreshDataFlag, TimeOutPeriod)
```

- Mother function (getpv)

- Daughter functions:
  - **getpvmodel** (function to be tuned for agiven facility)
  - **getpvonline** (function to be tuned depending on the controls choice)

Storage Configuration

- Definition of a directory tree for saving data
- Definition of Generic filename for Golden
- Naming convention for different types of measurements
  - BPM related,
  - Response matrices
  - Setpoint files
  - Etc.

BPMRespMat_2020-11-16_19-47-55.mat
BPMRespMat_2020-11-16_19-48-00.mat
BPMRespMat_2020-11-23_16-01-39.mat
BPMRespMat_2020-12-04_08-45-46.mat

- Golden File, Setpoints Management
  - Generic function to define them (*copybpmrespfile*…)
  - Backup (roll-back feature)

- StorageRingdata/BPM/
- StorageRingdata/BPM/Golden
- StorageRingdata/BPM/User

- StorageRingdata/BBA

- StorageRingdata/Response
- StorageRingdata/Response/BPM
- StorageRingdata/Response/Chro
- StorageRingdata/Response/Tune

- StorageRingdata/Response/LOCO

Response Matrix

1/ R = getbpmresp → default
2/ R = getbpmresp(Filename)
3/ R = getbpmresp(‘’) → open a dialogbox

Data Storage Policy and Configuration

Definition of a DATAROOT for storing all types of measurements

- Per default, all data are saved with structured format and metadata
  - Bring coherence and help the analysis
  - Information about the context
    - Active (status) actuators, Monitors
    - Device List
    - Operation Mode (Lattice type, setting)
    - Timestamp (ML based or control based)
    - Acquisition time
  - Other metadata can be added at will
    - Ring, LINAC, transfer line
    - Model, online, simulator
    - Stored beam
    - Data descriptor

```
            Data: [122×1 double]
      FamilyName: 'BPMx'
           Field: 'Monitor'
      DeviceList: [122×2 double]
          Status: [122×1 double]
            Mode: 'Simulator'
           Units: 'Hardware'
     UnitsString: 'mm'
  DataDescriptor: 'Horizontal Orbit'
       CreatedBy: 'getx'
               t: 0
            tout: 0.6341
        DataTime: [122×1 double]
       TimeStamp: [2024 6 18 21 3 59.1335]
```

BPM reading

```
Data
Monitor
Actuator
ActuatorDelta
GeV
TimeStamp
DCCT
ModulationMethod
WaitFlag
ExtraDelay
Units
UnitsString
DataDescriptor
CreatedBy
OperationalMode
```

BPM to dipole corrector response matrix

Selecting the Operational Mode

- Select a **model, a lattice** (storage ring: nominal optics, high chromaticity, low-alpha, etc.)
- Select Setting (e.g. filling patterns)

- Specific configuration
  - Lattice name (AT file)
    - Updating atindex
    - Updating s-location
    - Update names
  - RF step for dispersion, chromaticity measurements
  - Step size for response-matrices, etc.
  - Response Matrices (tune, orbit, chroma, etc.)
    - Model or experimental based
  - Calibration curve of magnets
  - Splitting of elements could be different
  - Golden parameters (tunes, chroma, etc.)
  - Feedback configuration (matrice, SVD threshold, etc.)
  - Cycling curve

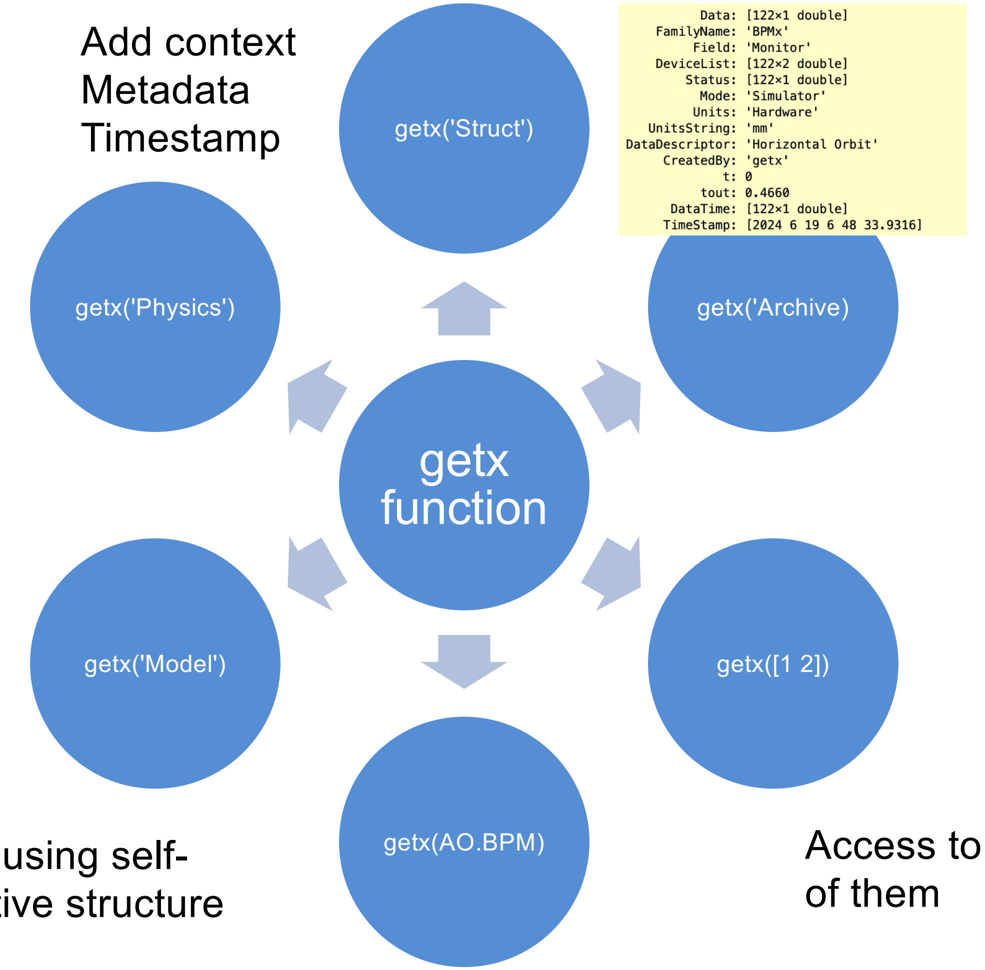
- Specific configuration
  - Magnet tolerances
  - Waiting Time (online)
  - Error handling
  - Status of element (disabling BPM, magnets)
  - Storage location
  - File extension
  - Reference Directory
  - Default units
  - Default mode

BPM: Getting Horizontal Position

Calibrated data (gain, roll)

Active BPMs: status = 1

Add context
Metadata
Timestamp



'Waiting Flag'
Quality Control

BPMdelay

Hardware units (µm)
Physics units (m)

Store Data in a file of the measured data ML file tree

Offset value (BBA)
Golden value

Online
Simulator (mimicking online)
Model (AT)

Access using self-descriptive structure

Access to a given BPM or a list of them

Modes of acquisition (TbT, Fast, slow, averaging, noise)
Access mode (normal, Special)

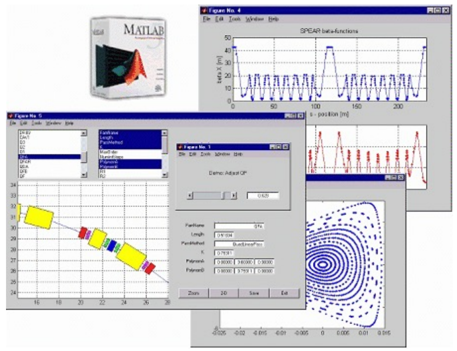
getx: alias to getam('BPMx')

ML and Model Interaction (AT), Making full use of the Families

- Simple input/output for ready-to-use data
  - modeltwiss(‘BPMx’)
  - modelbeta(‘BPMx’)
  - modeldisp(‘HCOR’)
  - modelphase(‘BPMy’)
  - modelbeamsize
  - modelchro
  - modeltune

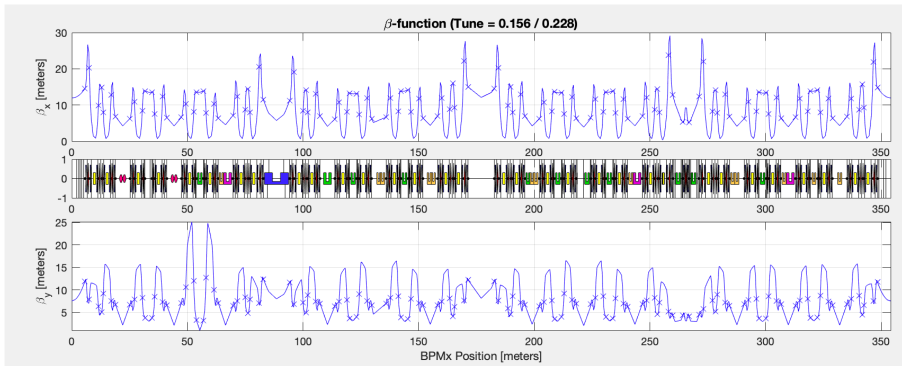
‘Display’ Flag
‘NoDisplay’ Flag

Accelerator Toolbox



https://atcollab.github.io/at/

Accelerator Synoptic



- Instead of low-level AT function
  - `[ringdata,elemdata]=atlinopt6(FODOcellSext,1:length(FODOcellSext)+1);`
    - `Format output`
      - `beta=cat(1,elemdata.beta);`
      - `spos=cat(1,elemdata.SPos);`

Library of Accelerator-Oriented Functions

There are hundreds of functions for accelerator control

- setorbit – general purpose global orbit correction function
- setorbitbump – general purpose local bump function
- settune – sets the storage ring tune
- setchro – sets the storage ring chromaticity
- measchro – measure the chromaticity
- measdisp – measure the dispersion function
- quadcenter, quadplot – finds the quadrupole center (BBA)
- physics2hw – converts between physics and hardware units
- measbpmresp – measure a BPM response matrix
- measlifetime – computes the beam lifetime
- minpv/maxpv – min/max value for family/field
- srcycle – standardizes the storage ring magnets
- scantune – scan in tune space and record the lifetime
- scanaperture – scans the electron beam in the straight sections and monitors lifetime
- finddispquad – finds the setpoint that minimizes the dispersion in the straight sections.
- rmdisp – adjusts the RF frequency to remove the dispersion component of the orbit by fitting the orbit to the dispersion orbit
- etc

- BPM Management
- Magnet Management

- Orbit factory
- Optics Factory

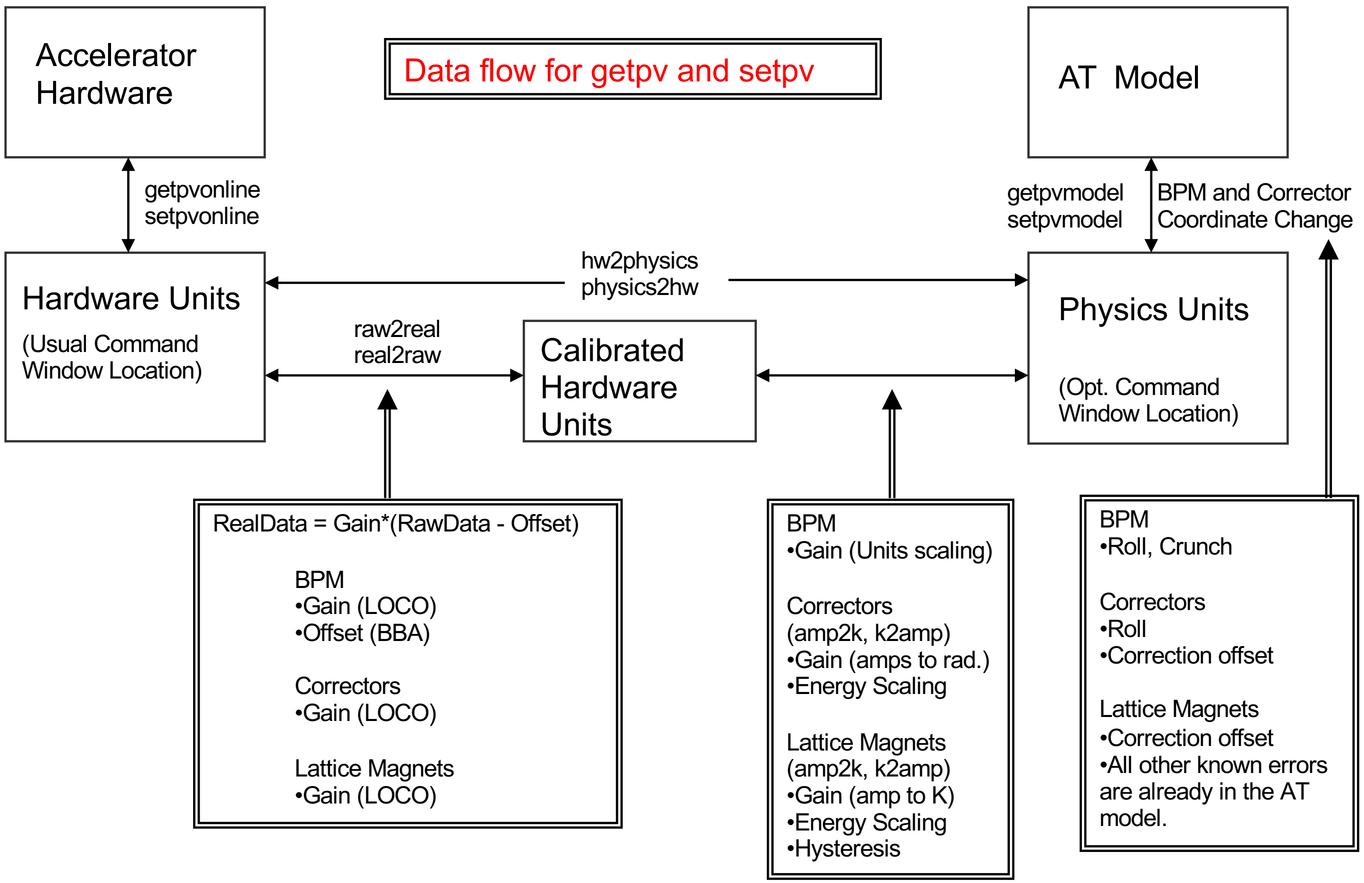
Highlights on a few Nice Features

- Overloading of the function name
  - Easy adaptation to a specific facility without breaking the legacy version
- Function Alias
  - Ease pre-configured function manipulation with well-known standard input argument
- Special function
  - Change/overload the generic behavior of the function
    - Example for BPM reading
      - Perform averaging
      - Access synchronized turb by turn data
      - Add extra tests to ensure a quality service level
- Group behavior
  - Changing several parameters together and linking variation between parameters
  - Introduce grouped variation (timing, synchronization based)
- Tagging family
  - Assigning family for generic application
- Error handling
- Limited intelligence
  - If a golden response matrix is not defined, the ML will propose to measure a new or to take an archived one

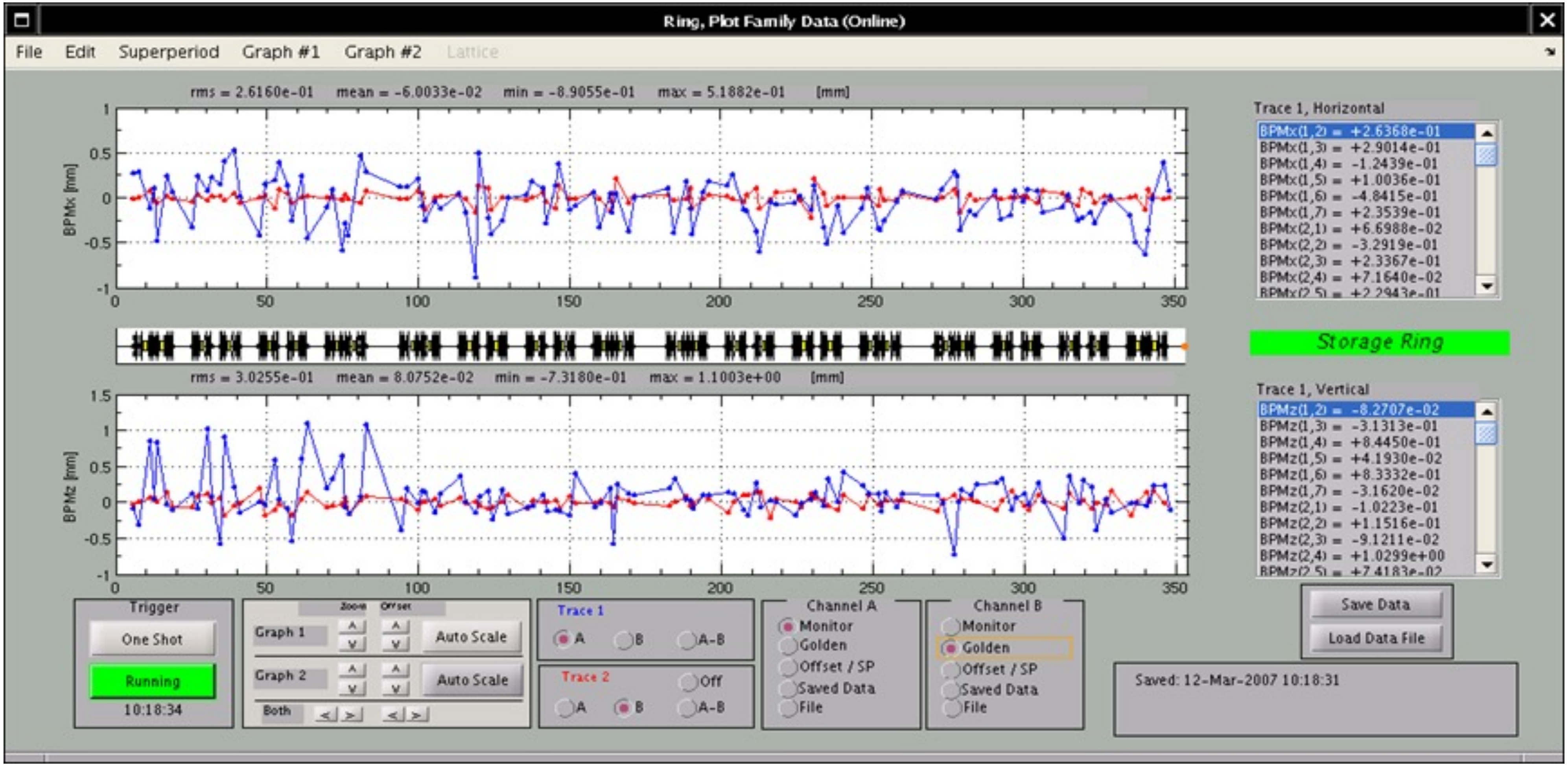
Fast Access to elements using ATindex

Middle Layer Data Flow Diagram

MML include a calibration model (BPM response, power supply physics2hardware units)

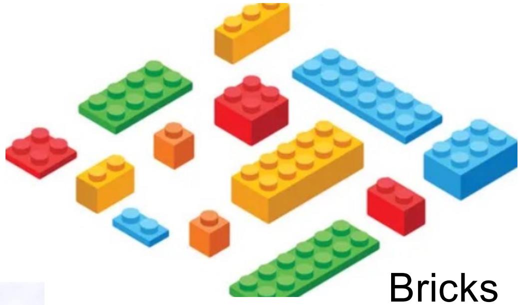


Generic Application to Plot/Compare Data for a given Family

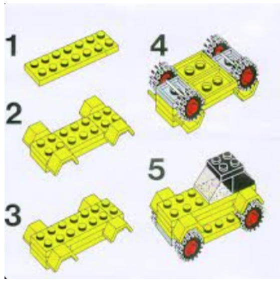


ML Direct Benefits

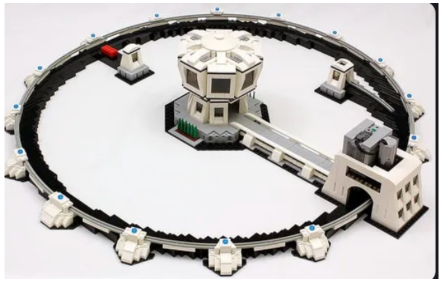
- Once configured, MML gives access for free to a full set of needed functions, applications that a bug-free and ready for use
- Ease of Access using Natural Language of Accelerator Physicists
- Easy to add a new lattice, mode of operation, to handle misbehaving equipment
  - A single-entrance point
- ML relies on a very modular approach
  - Normally only low-level parts of MML need modification and tuning
    - Control related
    - Model related
    - Configuration related
  - High-level applications do not need modifications besides a configuration part
- A consistent and homogeneous way to access and store the data
  - A function knows how to handle data
- Saving time and energy
  - Building trust and ease of use.
  - Do not reinvent the wheel focus on what matters the most
    - Everyone needs to measure the dispersion, the beta function, the close orbit, the chromaticity, etc.
- Library of shared accelerator-oriented functions up to GUIs
- Note that code genericity has a price in code performance



Bricks



Functions



Full Application

Thank you for your Attention

Questions

Important information SOLEIL / SOLEIL II upgrade (South of Paris, France)

- Permanent Position (Junior/Senior) open in the Accelerator Physics Group
- Permanent Position (Engineer/PhD) open in Operation Group