A draft document for discussion and brainstorming

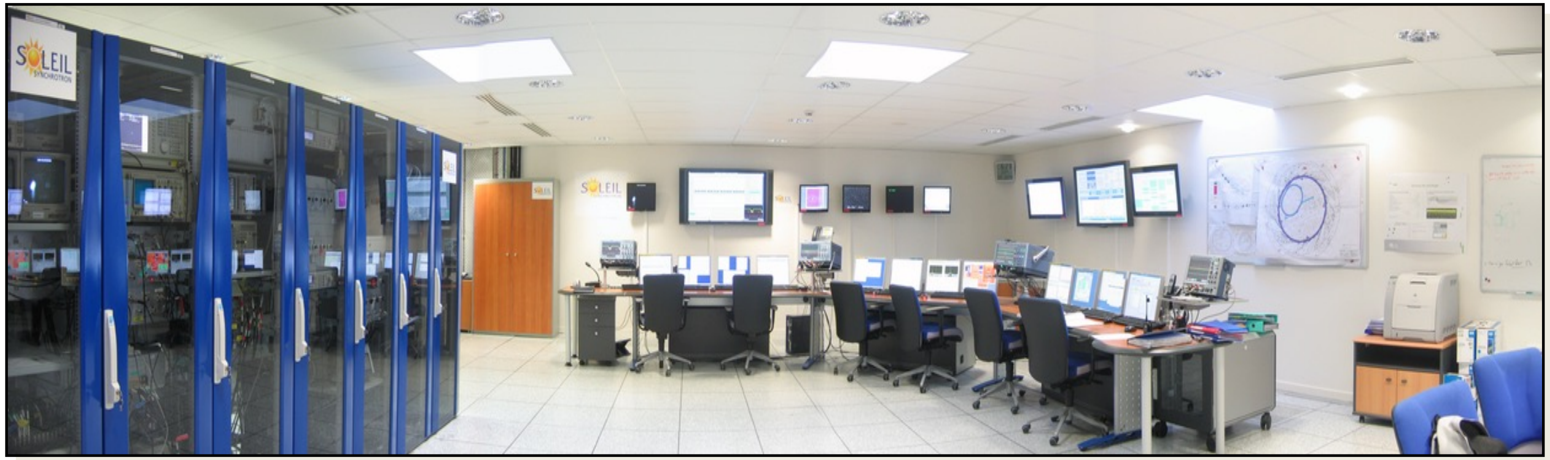
# Middle Layer (Matlab/Python) Discussion

Laurent S. Nadolski

# Outlook

- Introduction to MML (LEGACY) →GETTING the spirit of a middle layer
  - **The Matlab Middle Layer**
    - Extensive Definition
    - Examples
  - **Reasons for success and what has been learned …**
    - Choice of the tools
    - In house developments
    - TANGO cooperation and collaboration

- Open discussion about Middle Layer (ML)
  - Do we need a Python version? (pyML)
  - Do you need to refresh the Matlab version? (MML)



# Using Matlab for Accelerator Experimentation and Control
# or
# A Matlab "MiddleLayer" (MML)

Adapted slides by Gregory J. Portmann

Jeff Corbett, Andrei Terebilo, James Safranek (SSRL)
Christoph Steier, Tom Scarvie, Dave Robin (ALS)
Laurent S. Nadolski (SOLEIL)

# MML community around the word:

a non-exhaustive list

Many users, very few developers

**North America:** ALS, SSRL (SPEAR3), Duke FEL, NSLS2, (VUV or X-Ray rings), CLS, …

**Europe:** SOLEIL, LAL/THOMX (France), DIAMOND (UK), ALBA (Spain), KIT/ANKA (Germany), ILSF (Iran), MAX IV (Sweden), SOLARIS (Poland), IJCLAB (France), BESSY and HZB (Germany), PETRA-IV (Germany)…

**Asia:** PLS2 (Korea), SLS (Thailand), SSRF (China), NSRRC/TPS (Taiwan), …

**Middle East:** SESAME (Jordan)

**Australia:** ANSO

# Why MATLAB?

- Matrix programming language
  (variables default to a double precision matrix)
- Extensive built-in math libraries
- Active workspace for experimentation and algorithm development
- Easy of import/export of data
- Graphics library
- Compact code and good readability
- Adequate GUI capabilities
- Platform independents

# Automating Physics Experiments
(without becoming a software engineer)

## Goals

- Develop an easy scripting method to experiment with accelerators (accelerator independent)
  - Remove the control system details from the physicist (like Tango names and how to connect to the computer control system)
  - Easy access to important data (offsets, gains, rolls, max/min, etc.)
- Integrate simulation and online control. Make working on an accelerator more like simulation codes.
- Integrate data taking and data analysis tools
- Develop a software library of common tasks (orbit correction, tune correction, chromaticity, ID compensation, etc.)
- Develop high-level control applications to automate the setup and control of storage rings, boosters, and transfer lines.

# Matlab Toolbox Suite for Accelerator Physics

- MiddleLayer + High Level Applications
  1. Link between applications and control system or simulator.
  2. Functions to access accelerator data.
  3. Provide a physics function library.
- Control system flavor:
  - MCA, LabCA, SCAIII – Matlab to EPICS links
  - TANGO/Matlab binding
- Accelerator Toolbox for simulations
- LOCO – Linear Optics from Closed Orbits (Calibration)
- NAFF Library (frequency maps)
- SC, etc.
- Python…
- Used for transfer lines, Booster, Storage Ring

# Software Interconnection Diagram

**Replace TANGO by your control system**

Hiden to users

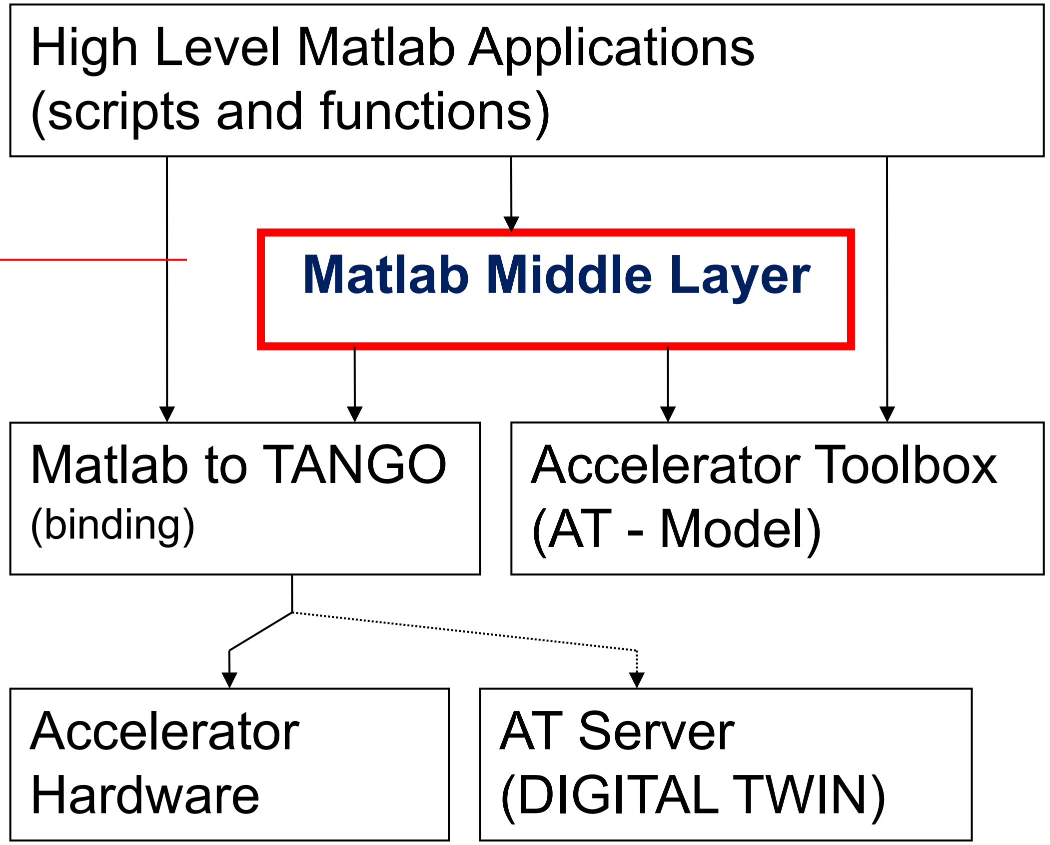
- TANGO system
- TANGO naming convention

Device name:

ANS-C01/DG/BPM.2

Attribute name:

XPosSA



# Function Library

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
- 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

# Data Management

- Beam Position Monitors
  – Attribute names, gains, roll, crunch, offsets, golden, standard deviations
- Magnets
  – Attribute names, gains, offsets, roll, setpoint-monitor tolerance, amp-to-simulator conversions, hysteresis loops, max/min setpoint
- Other equipment: Vacuum, loss monitors, etc.
- Response matrices (Orbit, Tune, Chromaticity)
- Lattices (Save and restore)
- Measurement archiving
  – Dispersion, tunes, chromaticities, quadrupole centers, etc.
- TANGO configuration
  – Device & attribute properties
  – Historical data archiving
  – …

# Matlab feedbacks at SOLEIL

- Relatively easy to use. Most people start writing useful scripts in a few hours.
- MiddleLayer + LOCO + AT + TANGO cover many of the high-level software concerns for storage rings. Hence, not every accelerator has to spend resources coding the same algorithms.
- Thousands of dedicated accelerator hours have been spent testing, improving, debugging, and exercising the middle-layer software.
- 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.
- Easy way for prototyping high-level control applications
- The semi-machine independence software has fostered collaboration and code sharing between the laboratories.

# Direct Benefits

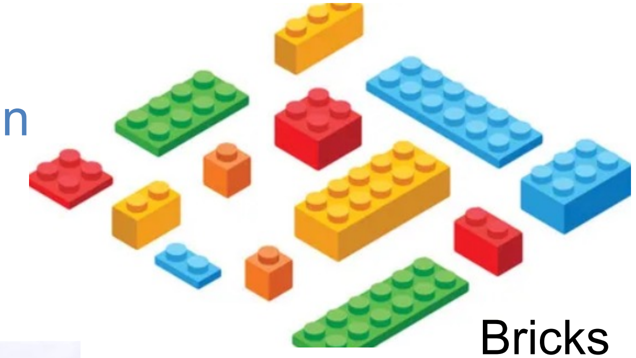
- Once configured, MML gives access for free to a full set of needed function bug-free and ready for use

- Saving time and energy

- Example of a higher level of abstraction wrt to AT function
  - To a full set higher-level script/applications than AT
    - [beta_x, beta_y] = modelbeta instead of
      - [output, ~, ~] = atlinopt(lattice, 0, 1:length(lattice))
      - beta_x = arrayfun(@(a).a.beta(1), output);
    - [beta_x, beta_y] = modelbeta('BPMx') apply a mask for BPM location only

- Building trust and ease of use.

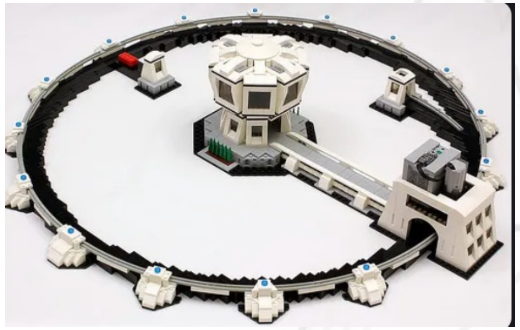
- Do not reinvent the wheel focus on what matters the most
  - Everyone needs to measure the dispersion, the beta function, the close orbit, the chromaticity, etc.



Bricks

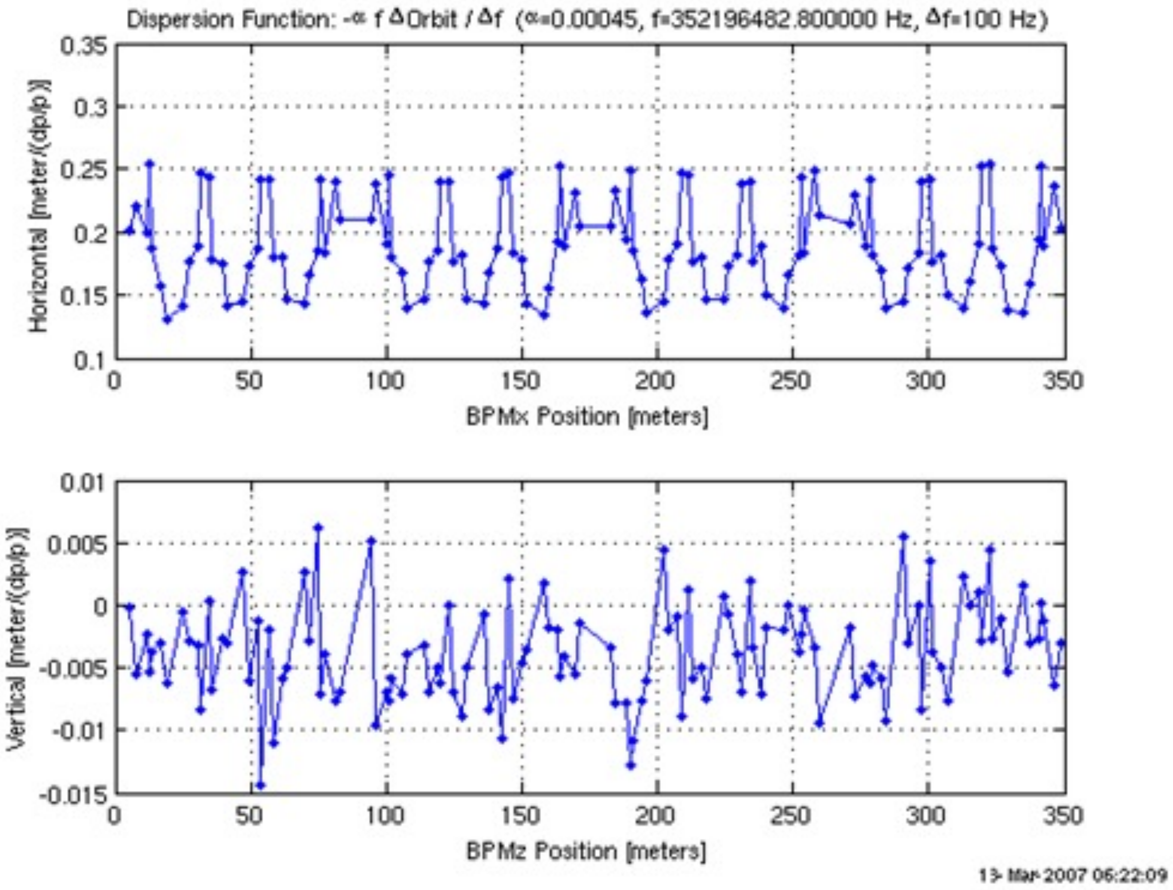

functions

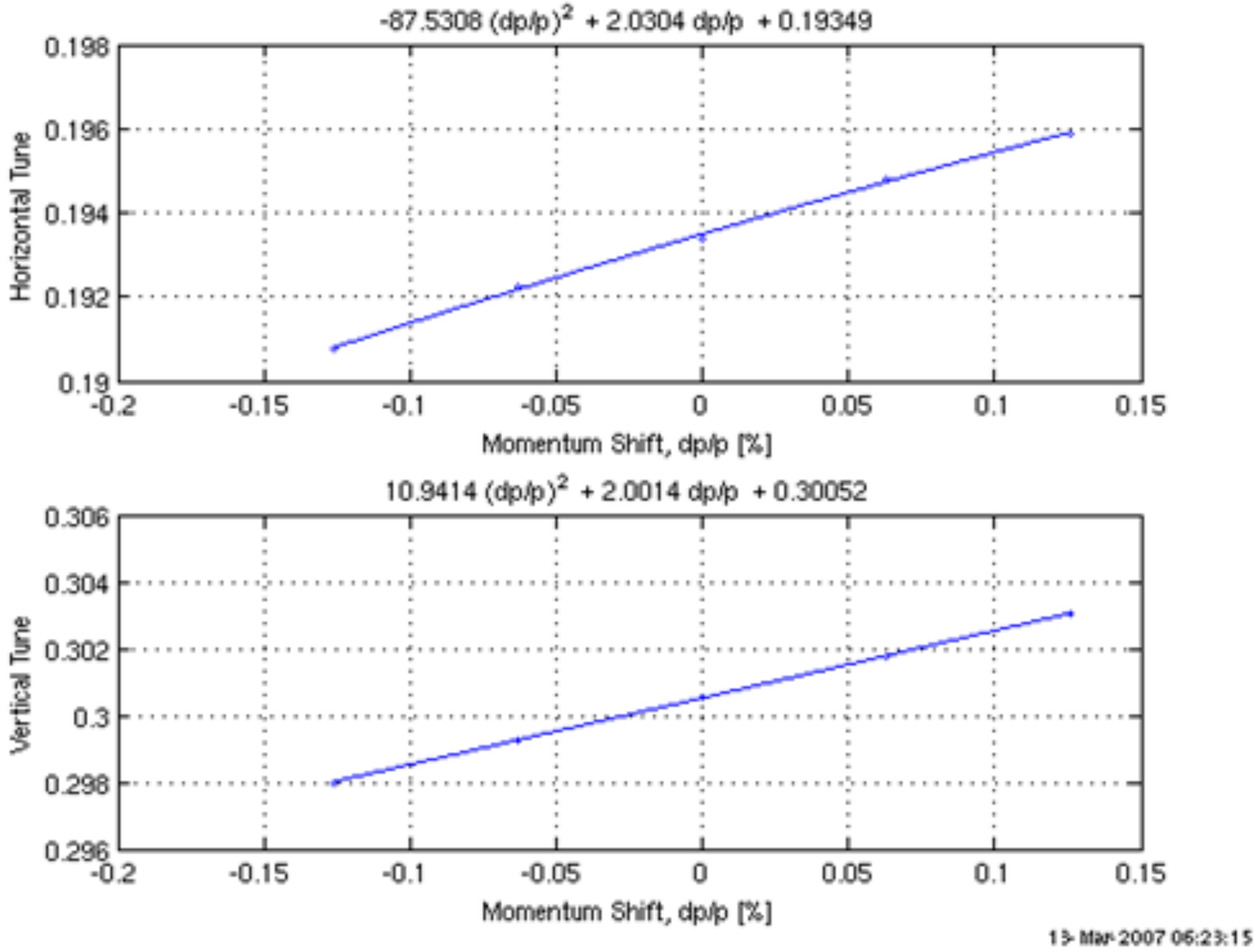
https://shorturl.at/lmn78



Full application

# Dispersion/Chromaticities

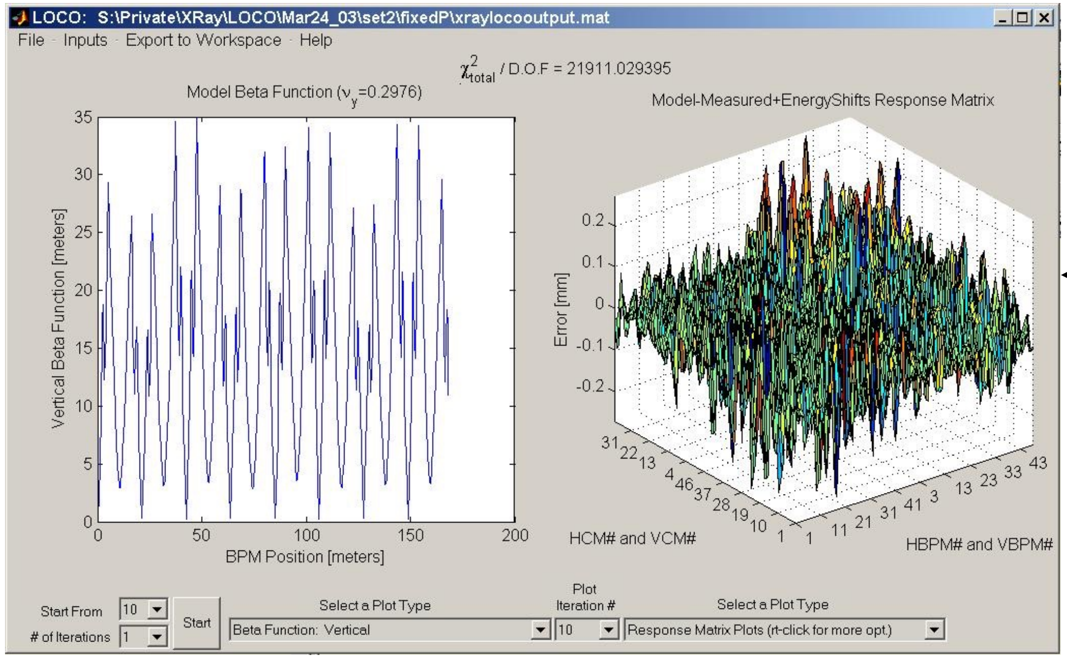



MML core functions

# LOCO Optics Analysis

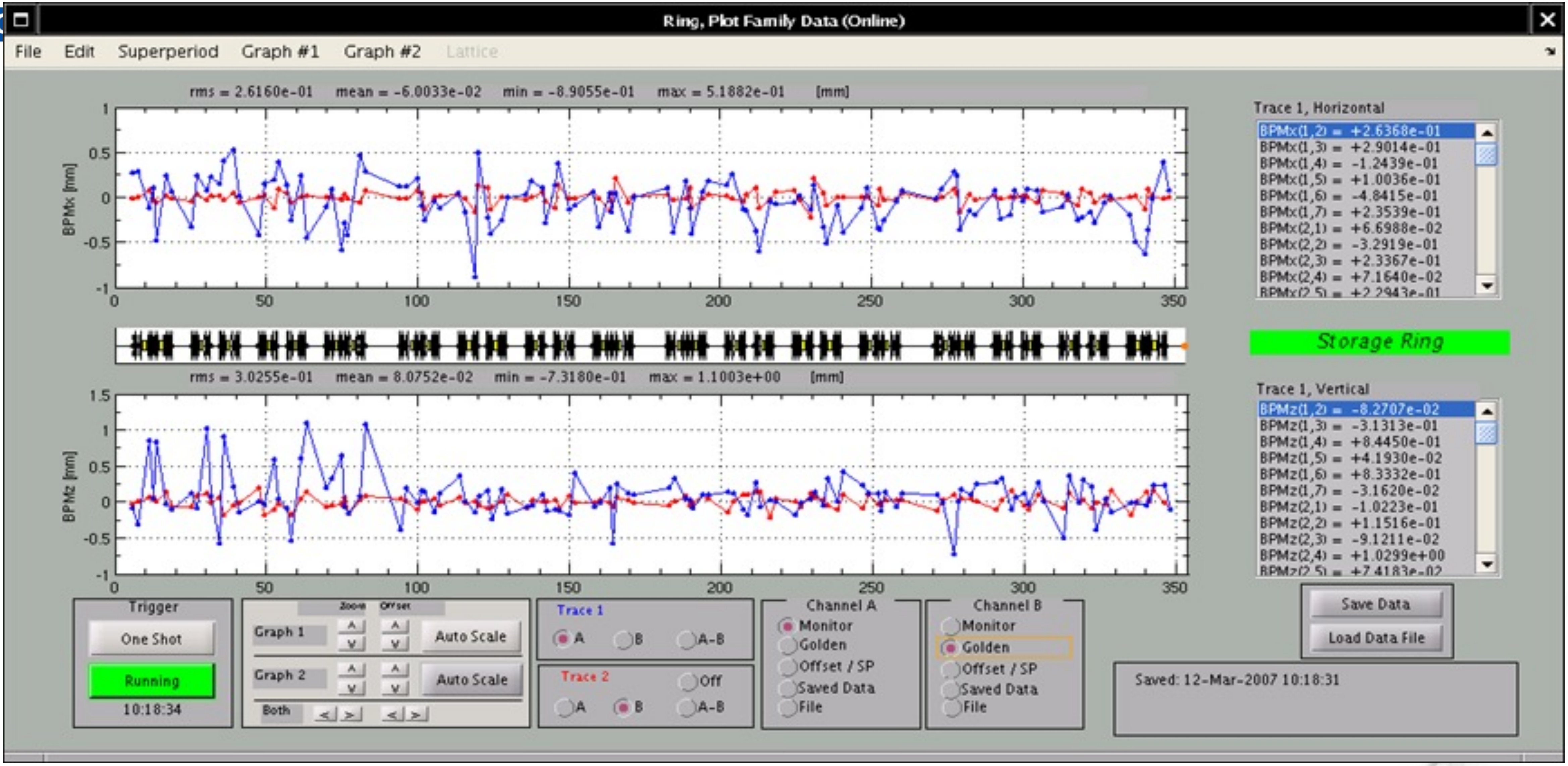
- Calibrate/control optics using orbit response matrix
- Determine quadrupole gradients
- Correct coupling
- Calibrate BPM gains, steering magnets
- Measure local chromaticity and transverse impedance




New MATLAB version of code

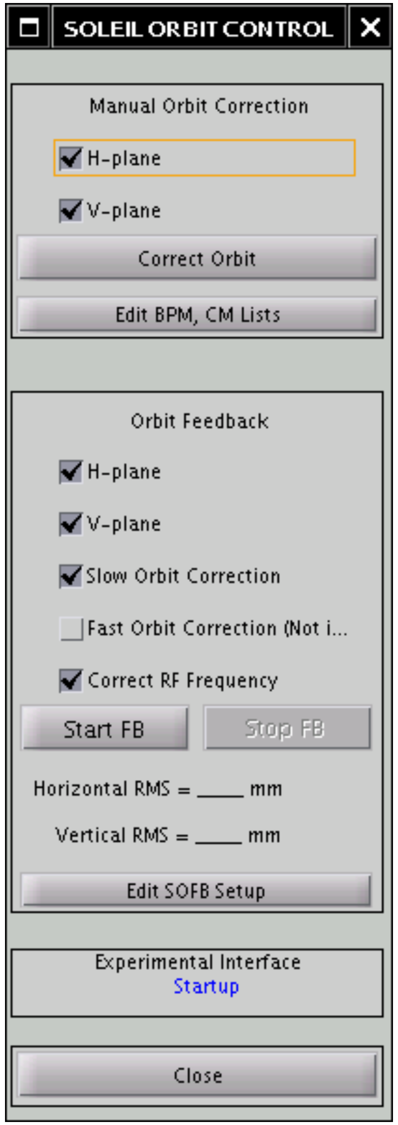
- Rewritten from FORTRAN
- Linked to control system
- Linked to AT simulator

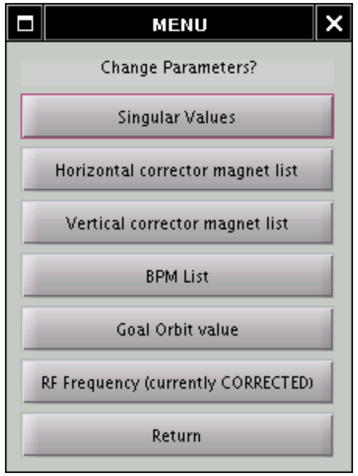
# Displaying closed orbit



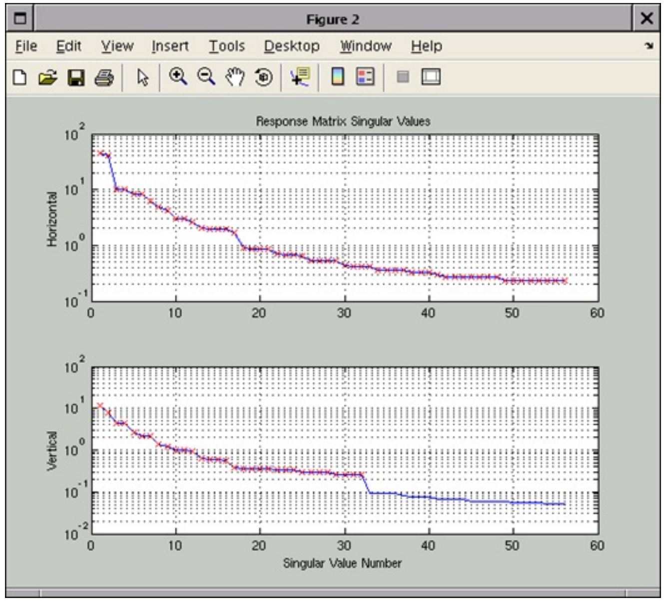

Upgraded from ALS

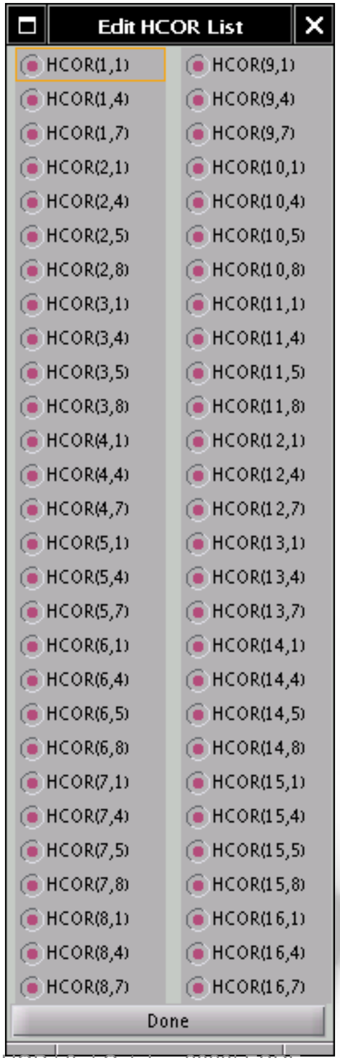
# Orbit Correction, SOFB




1. $\Delta X = X_{lu} - X_{ref}$
2. $\Delta I = R^{-1}\, \Delta X$
3. $I = I_0 + \Delta I$



# Discussion

# Feedback from the community

- **Control System**
  - What control system do you use in your control room (EPICS, TANGO, other)?

- **Matlab Middle Layer (MML)**
  - What is the status in your lab?
  - What are your use cases: simulation? operation? commissioning? beam-based measurements during machine days?
  - What do you want to improve?
  - What is your timetable and what are the available resources of your lab for working on the middle layer?

- **Python Middle Layer (PML)**
  - How mature is PML?
  - Is there a strategy for moving between Matlab and Python?
  - What barriers have been identified?
  - What is the status of Python integration in your lab and control room?
  - What are your use cases: simulation? operation? commissioning? beam-based measurements during machine days?
  - What is your timeline and what are the available resources of your lab for working on the middle layer?

# Feedbacks

- Laboratory Feedback
  - ALBA, IJCLAB, BNL, ELETTRA, BESSY, HBZ, MAX IV, SOLEIL, PETRA IV, ESRF, ALS

- Many use cases still in Matlab (legacy)
  - Many lab cannot afford leaving Matlab on the short term

- Strong interest in Python-based Middle-layer

- Strong interest in Python-based Digital Twin

- Strong will to migrate to Python in the medium and long term

- Little or no resources

- No clear roadmap in many labs

Many versions and local development (branches)

**MML very successful** but not released anymore

Need a global effort a community Collaborative development

# An enlightening extended Feedback (by Pierre S.)

- *I think that its a very good idea to foster this collaboration. Personally, I see it also a window of opportunity:  in my opinion, these united forces could allow the reorganizing and modernizing what exists.*

- *I'd like to suggest*
  - *to consider to reorganize code bases to express the inherent structure a bit clearer to newcomers*
  - *review to which extent modern language constructs would simplify implementation or pave the path to further applications*
  - *review solutions existing elsewhere and adopt best practice.*

- *In case there is interest I am happy to elaborate on these points. Please find below some short keywords that would give an idea.*

# Suggestion (Continue)

- Codebase reorganization (some suggestions below)
  - AT split up to
    - Common: c-integrators used by Matlab and python
    - AT? For Matlab part
    - pyAT for python parth
  - ML split  up
    - Control system communication
    - Engineering to physics
    - Application layer (loco etc.)
- Projects to look as sources that could inspire what to use
  - ML
    - Functional mockup interface, open simulation platform. iso standard on digital twinning
    - Bluesky software stack

•Points that could help structure the discussion

- Requirements for modeling and measuring steady state versus transient processes
- Common ontology and data models, abstraction used
  - for expressing device data and device characteristics (expected timeouts, settle times, configuration)
  - for handling the parts that can run asynchronously and synchronously transparently
- Layers & Components:
  - analysis, measurement plans, device abstractions
  - patterns to applied
- Data exchange with databases or product lifecycle management systems next to their abstractions

# How to Improve MML

- Written 30 years ago, this is true, but it is fully functional

- Improvement needed
  - Get benefits from the MATLAB evolution
    - Code improvement
    - Performance improvement
    - Obsolescence handling
      - Function, interface
      - New graphical interfaces

  - Get benefits from the atcollab version AT

- Refactoring may be needed

- Regression test, unitary test (Matlab can do it also very well)

- For all of that. A Project is needed with resources, developers, testers…

# No Middle Layer in Python yet!

- Do we need a pyML (python middle layer to ease or life) and continue work in the spirit of MML?
- Extending the library of accelerator-oriented functions
- Tuning pyAT to your accelerator facility specificities in a more generic way

# Appendices

THERING

FAMLIST

AO

AD

## 6.1 Accelerator Object (AO)

| | |
|---|---|
| But | Information permettant la communication entre les familles et le système de contrôle/commmande |
| Lieu de stockage | Espace de travail de Matlab |
| Get/Set | *getfamilydata / setfamilydata* |

*Tableau 6.1: Accelerator Object*

## 6.2 Accelerator Data (AD)

| But | Variables liées au MML |
|---|---|
| Lieu de stockage | Espace de travail de Matlab |
| Get/Set | *getfamilydata / setfamilydata* |
| AD.Machine | nom de la machine, eg. 'ALS' ou 'SOLEIL' |
| AD.Directory.DataRoot | Racine de l'arborescence des fichiers de sauvegardes |
| AD.OpsData.RespFiles | Tableau de cellules des fichiers de matrices réponses, eg. {'respmatbpm_08-06-2002', 'respmattune'} |
| AD.ATModel | Nom de la maille AT |
| AD.BPMDelay | Temps d'attente entre deux relectures des BPM (attendre que les données soient renouvelées) |
| AD.TUNEDelay | Temps de delai pour les nombres d'onde (cf. BPM) |

SOLEIL SYNCHROTRON UPGRADE

## données

AcceleratorObject.(FamilyName)

| Champ | Description |
|---|---|
| FamilyName | - Nom de la famille ('BPMx', 'HCOR', etc.) (unicité requise) |
| FamilyType | - Nom de la catégorie d'éléments, par exemple 'QUAD' |
| MemberOf | - Tableau de cellules, par exemple {'QUAD', 'Magnet'} |
| Status | - 1 pour statut valide, 0 pour invalide |
| DeviceList | - Vecteur colonne [1 1; 1 2; 2 1, ...] |
| ElementList | - Vecteur colonne [1;2;3; ..; n] |
| Desired | - Structure (cf. infra) |
| Monitor | - Structure (cf. infra) |
| Setpoint | - Structure (cf. infra) |
| DeviceNames | - Matrice de cellules pour le nom du device Tango |

| Champ | Description |
|---|---|
| Position | - Vecteur colonne avec la position longitudinal le long de l'anneau (mètres) |
| AT | - Structure pour le simulateur AT (facultatif) |
| Golden | - Structure avec les valeurs de référence (facultatif) |

*Tableau 4.1: Champs d'une famille du MML.*

## Equipment Family (>> showfamily)

Bending Magnet – BEND
Quadrupoles – Q1, Q2, … Q11
Sextupoles – S1, S2, … S12
Quadrupoles tournés- QT
Correctors – HCOR, VCOR
Beam Position Monitors (BPM) – BPMx and BPMz
Others - RF, DCCT, TUNE, GeV

## Fields

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

# Scripting example: Orbit Correction

```
% Gets the vertical orbit
X = getam('BPMx');

% Gets the horizontal response matrix from the model
Rx = getrespmat('BPMx', 'HCM');       % 120x56 matrix

% Computes the SVD of the response matrix
Ivec = 1:48;
[U, S, V] = svd(Rx, 0);

% Finds the corrector changes use 48 singular values
DeltaAmps = -V(:,Ivec) * S(Ivec,Ivec)^-1 * U(:,Ivec)' *  X;

% Changes the corrector strengths
stepsp('HCM', DeltaAmps);

(setorbitgui done this more elegantly)
```

# Basic Calling Syntax

**Naming Convention: practical, easy to remember, …**

Family = Group descriptor (text string)
Field = Subgroup descriptor (text string)
DeviceList = [Sector Element-in-Sector]

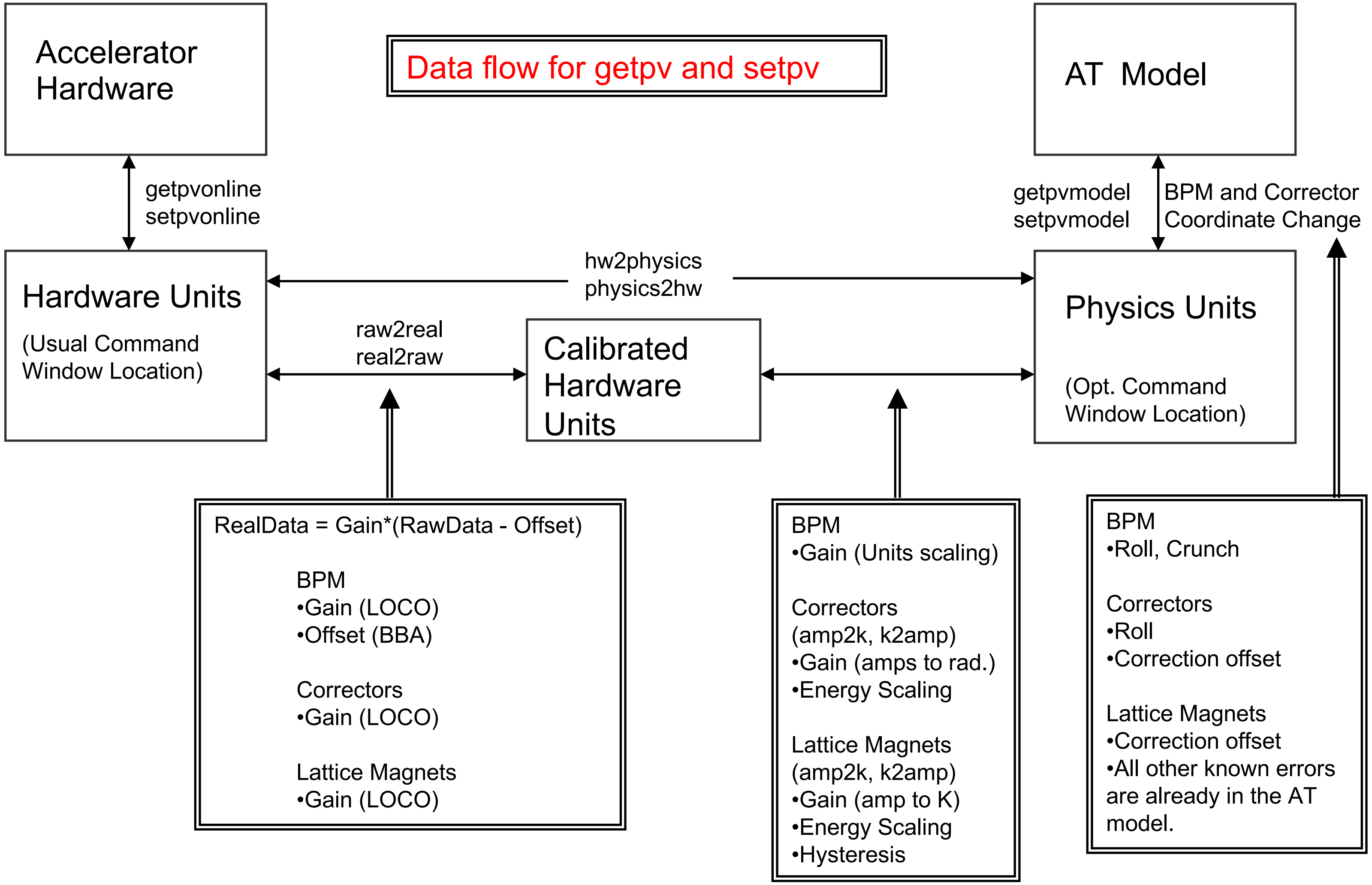
**Basic Functions**

```
getpv(Family, Field, DeviceList);
setpv(Family, Field, Value, DeviceList);
steppv(Family, Field, Value, DeviceList);
```

**Examples:**

```
x = getpv('BPMx', 'Monitor', [3 4;5 2]);
h = getpv('HCM', 'Setpoint', [2 1;12 4]);
setpv('QF', 'Setpoint', 81);
```

# MiddleLayer Data Flow Diagram



# Commands

- **Nomenclatures**
  - TANGO cf document, ex: ANS-C01/DG/BPM.2
  - Matlab Middle Layer
    - BPM [1 2] : BPM 2 of cell 1
  - *getam('BPMx',* [1 2]*)*
  - *family2tangodev('BPMx',[1 2])*
- **RF-Fréquency**
  - *getrf* : 352.1962246 MHz
  - 10 Hz, *steprf(10e-6)*
  - *getrf* : 352.1962346 MHz

# Controls: Real or Simulated Equipment?

- Make Model as default
  >> switch2sim

- Make Real Equipment as default
  >> switch2online

- Overloading commands:
  'Model' :
  getsp('HCOR', 'Model' );
  'Online'
  getsp('HCOR', 'Online' );

# Physics Units vs Hardware Units (SI)

- Make Hardware Units per Defaults

  ```
  >> switch2hw
  ```

- Make Physics Units per Default

  ```
  >> switch2physics
  ```

- Overloading Commands

  'Hardware' – Force hardware units for this function.

  'Physics' – Force physics units for this function.

Examples :

```
>> Amp = getpv('Q1', 'Hardware');
>>   K  =  getpv('Q1', 'Physics');
```