# Introduction to Matlab Middle Layer

Xiaobiao Huang

Based on an earlier version by Christoph Steier and Greg Portmann.

USPAS Jan. 2019 – Beam Based Diagnostics

# Matlab Middle Layer (MML)*

- Goals of MML: automating accelerator physics experiments
  - Develop an easy scripting environment to experiment with accelerators (accelerator independent)
    - Remove the control system details from the physicist (like channel 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.
  - 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 a storage ring.

*G. Portmann, J. Corbett, A. Terebilo, An accelerator control middle layer using Matlab, PAC'05

# History of MML

- Development of Matlab Middle Layer started >17 years ago at ALS
  - Greg Portmann, Winfried Decking, David Robin, Christoph Steier
- Other accelerators had similar requirements and went similar (or somewhat different) routes
  - APS – Tcl/TK
  - ELSA – EPOS and later Matlab
  - DESY – Matlab
- Later on AT (Terebilo, et al.) and LOCO (Portmann and Safranek) were ported to Matlab (from pascal and fortran) and combined with middle layer and many controls interfaces other than EPICS were added
- Now very widely used at more than ½ of 3rd generation light sources and some colliders (in some cases only LOCO, not full middle layer)

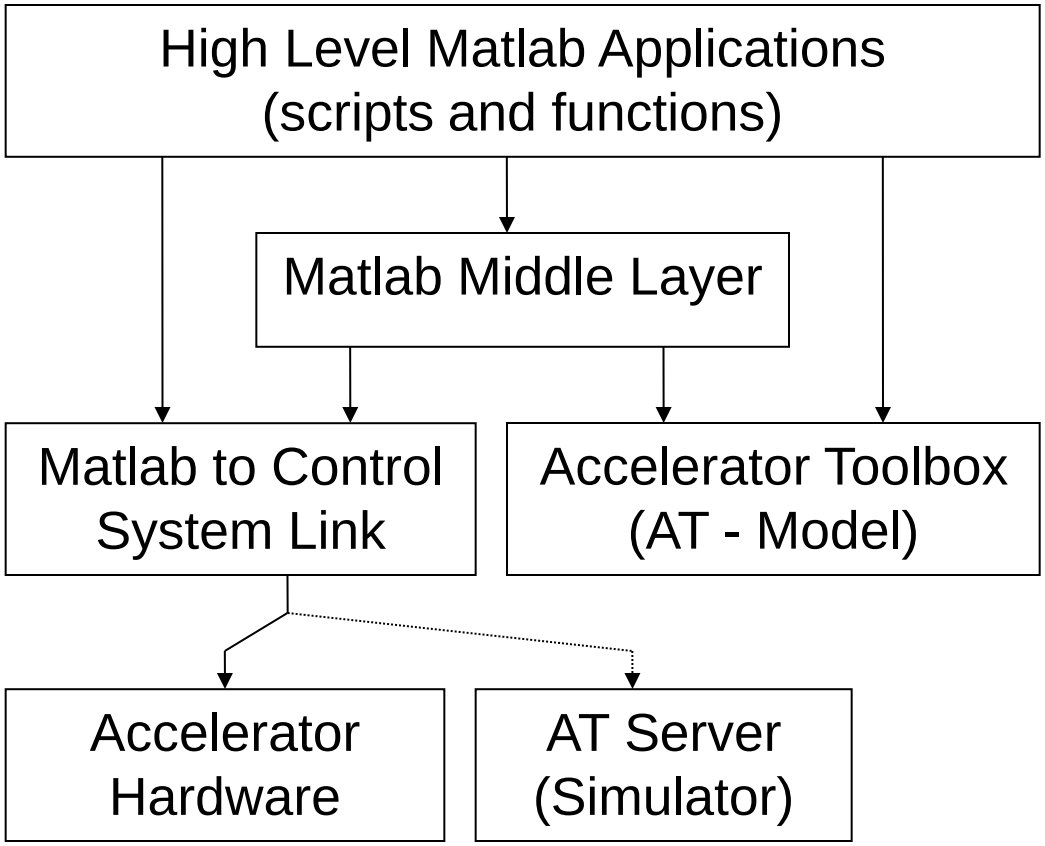
# Why Matlab

- Matrix programming language
- Extensive built-in math libraries
- Active workspace for experimentation and algorithms development
- Easy of import/export of data
- Graphics (easy presentation of data)
- Compact code and good readability
- Adequate GUI capabilities
- Platform Independent

# Matlab Toolbox Suite for Accelerator Physics

- MiddleLayer
  1. Link between applications and control system or simulator.
  2. Functions to access accelerator data.
  3. Provide a physics function library.
- High Level Applications
  - Within the standard MML distribution
  - Additional applications that are built upon MML.
- LabCA, MCA, SCAIII – Matlab to EPICS links
  - Other control systems can be linked, too.
- AT – Accelerator Toolbox for simulations
- LOCO – Linear Optics from Closed Orbits (optics calibration)

# Software Interconnection Diagram



# Basic calling syntax for accessing the control system

## Naming Convention

Family = Group descriptor (text string)
Field = Subgroup descriptor (text string)
DeviceList = [Sector Element-in-Sector]
ElementList = [Element-in-Family]
ChannelName = Control System name (text string)

## Basic Functions

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

## Examples:

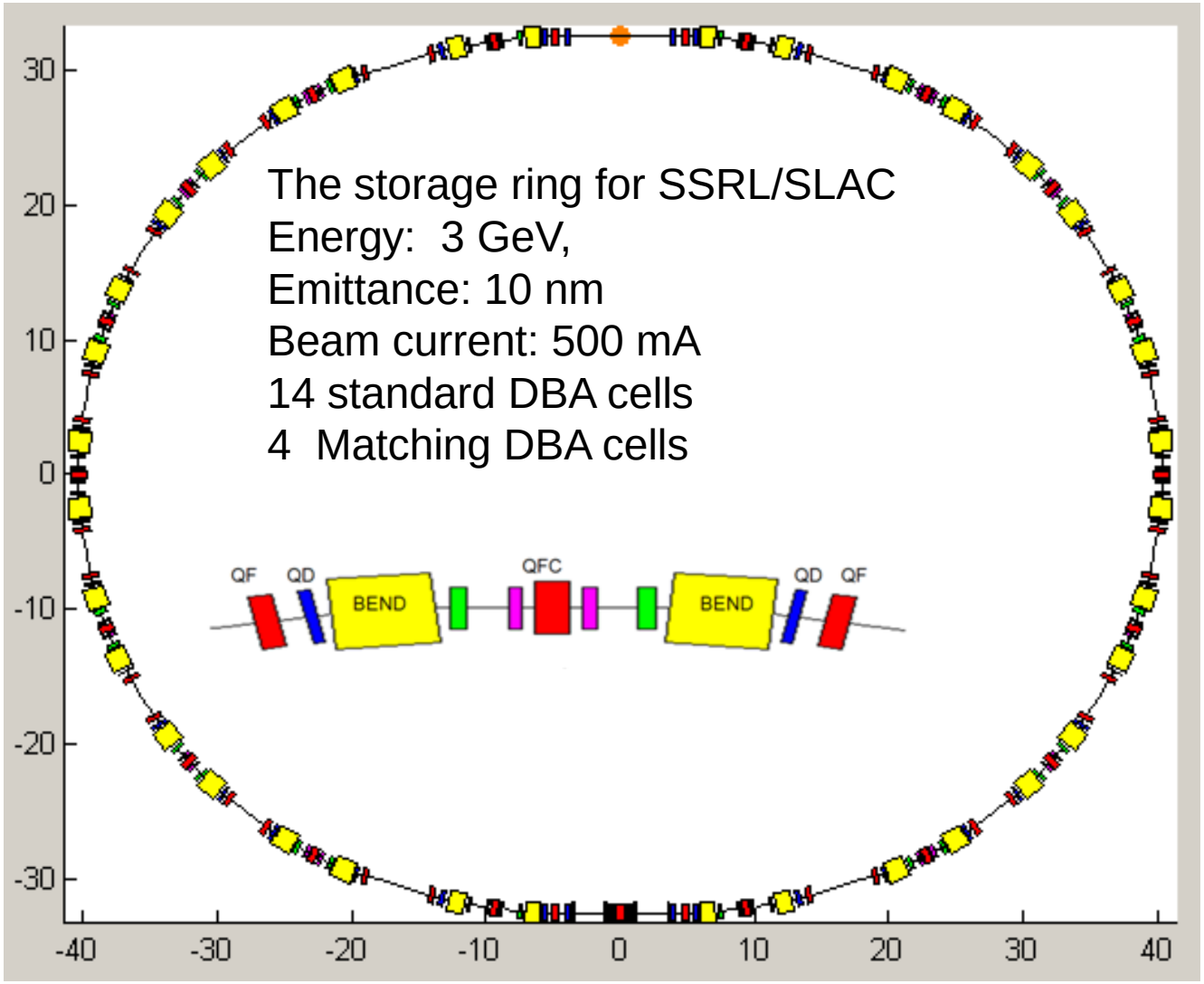
x = getpv('BPMx', 'Monitor', [3 4;5 2]);
x = getpv('BPMx', [3 4;5 2]);
x = getpv('BPMx', [17;27]);
h = getpv('HCM', 'Setpoint', [2 1;12 4]);
setpv('QF', 'Setpoint', 81);

'03G-BPM4:U'
'05G-BPM2:U'

'02G-COR1H:CurrSetpt '
'12G-COR4H:CurrSetpt '

There are many shortcut functions: getam, getsp, setsp, getbpm, getx, gety, getrf, getdcct, gettune, etc

# The SPEAR3 storage ring (used in examples)



# SPEAR3 Family Names

## Families

Bend magnets – BEND, BDM
Quadrupoles – QF, QD, QFC,Q[DF][XYZ],Q9S
Sextupoles – SF, SD, SFM, SDM
Skew quadrupoles - SkewQuad
Correctors – HCM, VCM
Beam position monitors – BPMx and BPMy
Other - RF, DCCT, TUNE, etc

## Fields

Setpoint, Monitor, Status, CommonNames, DeviceList, ElementList, Position, etc…

### Getting around with families, device lists, channel names

channel2dev – convert channel names to device list
channel2family – convert family to channel names
dev2elem – convert element list to device list
elem2dev – convert device list to element list
family2channel – convert family to channel names

# The database structures

- The Accelerator Object (AO) ---- getao
  - Store family information related to the control system
  - Get/Set: *getfamilydata* / *setfamilydata*

- The Accelerator Data (AD) ---- getad
  - Store Middle Layer setup variables
  - Get/Set: *getfamilydata* / *setfamilydata*

- The Physics Data (PhysData)
  - Store physics related data (in a file)
  - Get/Set: *getphysdata* / *setphysdata*

AcceleratorObject.(Family)

| | |
|---|---|
| FamilyName: | Family Name ('BPMx', 'HCM', etc.) (must be unique) |
| MemberOf: | Cell array of strings, for instance {'MachineConfig'; 'PlotFamily'; 'QUAD'; 'Magnet'} |
| Status: | 1 for good status, 0 for bad status |
| ElementList: | Column vector |
| Monitor: | Structure shown below |
| Setpoint: | Structure shown below |
| CommonNames: | String matrix of common names |
| Position: | Column vector of longitudinal position along the ring [meters] |
| AT: | Structure for the AT simulator (optional) |

# Switch between Model or Online Accelerator

- Make the model the default

  >> switch2sim
- Make the accelerator the default

  >> switch2online
- Mixed mode – use keyword overrides

  'Simulator' – Run the same code as online just use the AT model for input/output.

  'Model' – Some code uses the AT model more directly (like measbpmresp or measchro)

  Note: ‘Model’ and ‘Simulator’ are often the same.

  *example: getam('BPMx',[1 1], 'Physics', 'Model')*

# Switch Between Hardware and Physics Units

Hardware units: the units used in the control system (engineering units), such as ampere, MHz, mm.
Physics units: the units used in accelerator physics (models), such as radian, 1/m^2 (K1 for quads).

- Make the hardware units the default

  >> switch2hw
- Make the physics units the default

  >> switch2physics
- Mixed mode – use keyword overrides

  'Hardware' – Force hardware units for this function.

  'Physics' – Force physics units for this function.

  Example:

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

# Machine Physics Functions

There are hundreds of functions for accelerator control, monitoring and data analysis

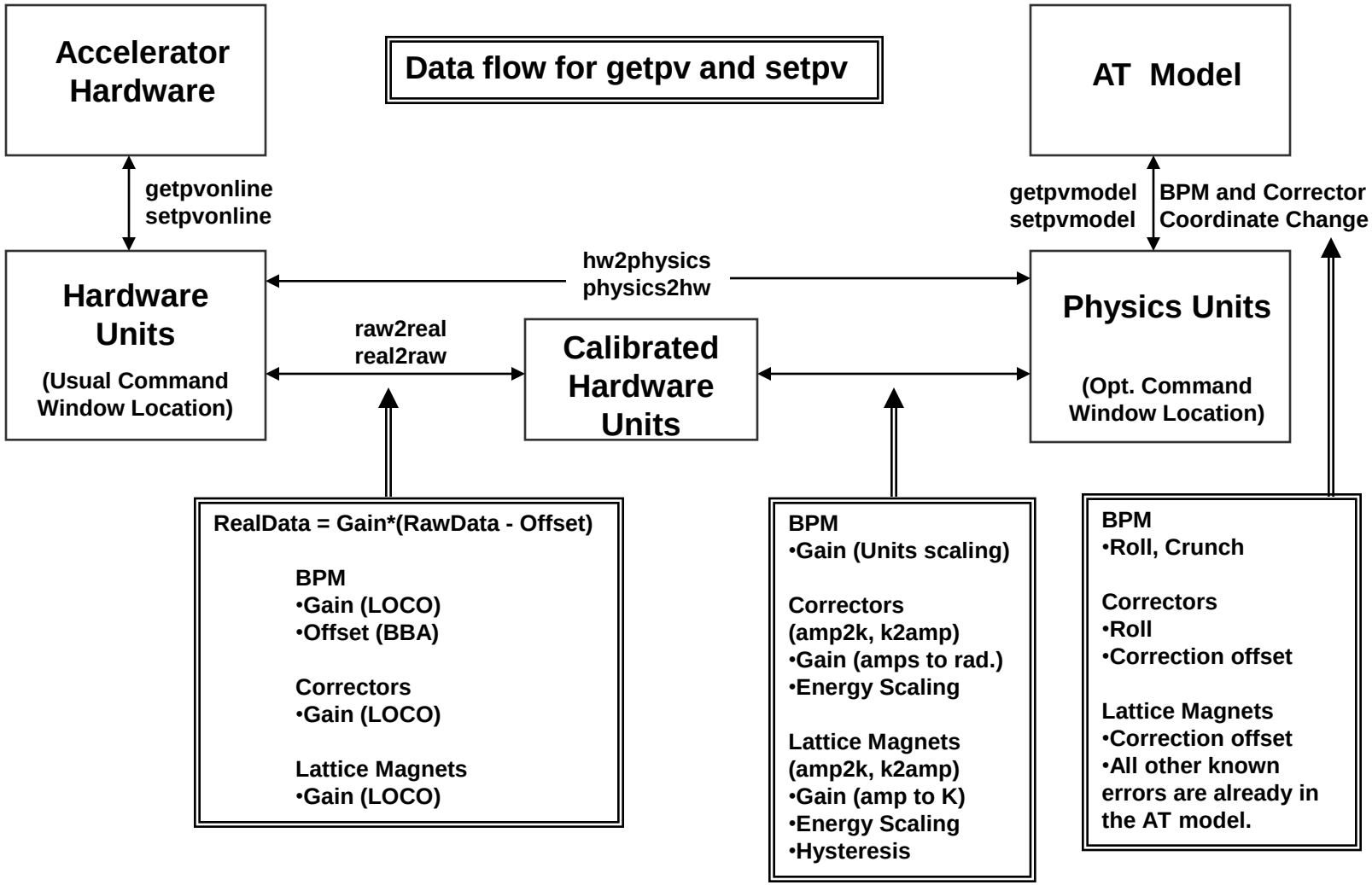
- setorbit – general purpose global orbit correction function
- setorbitbump – general purpose local bump function
- settune – sets the storage ring tune
- setchro – sets the storage ring chromaticity
- measchro – measure the chromaticity
- measdisp – measure the dispersion function
- quadcenter, quadplot – finds the quadrupole center
- physcis2hw – 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
  - Channel names, gains, roll, crunch, offsets, golden, standard deviations
- Magnets
  - Channel names, gains, offsets, roll, setpoint-monitor tolerance, amp-to-simulator conversions, hysteresis loops, max/min setpoint
- Response matrices (Orbit, Tune, Chromaticity)
- Lattices (Save and restore machine configurations)
- Measurement archiving
  - Dispersion, tunes, chromaticity, quadrupole centers, response matrices, etc.

Data are saved in pre-specified locations. When operational data files are updated, old files are backed up with time stamps.

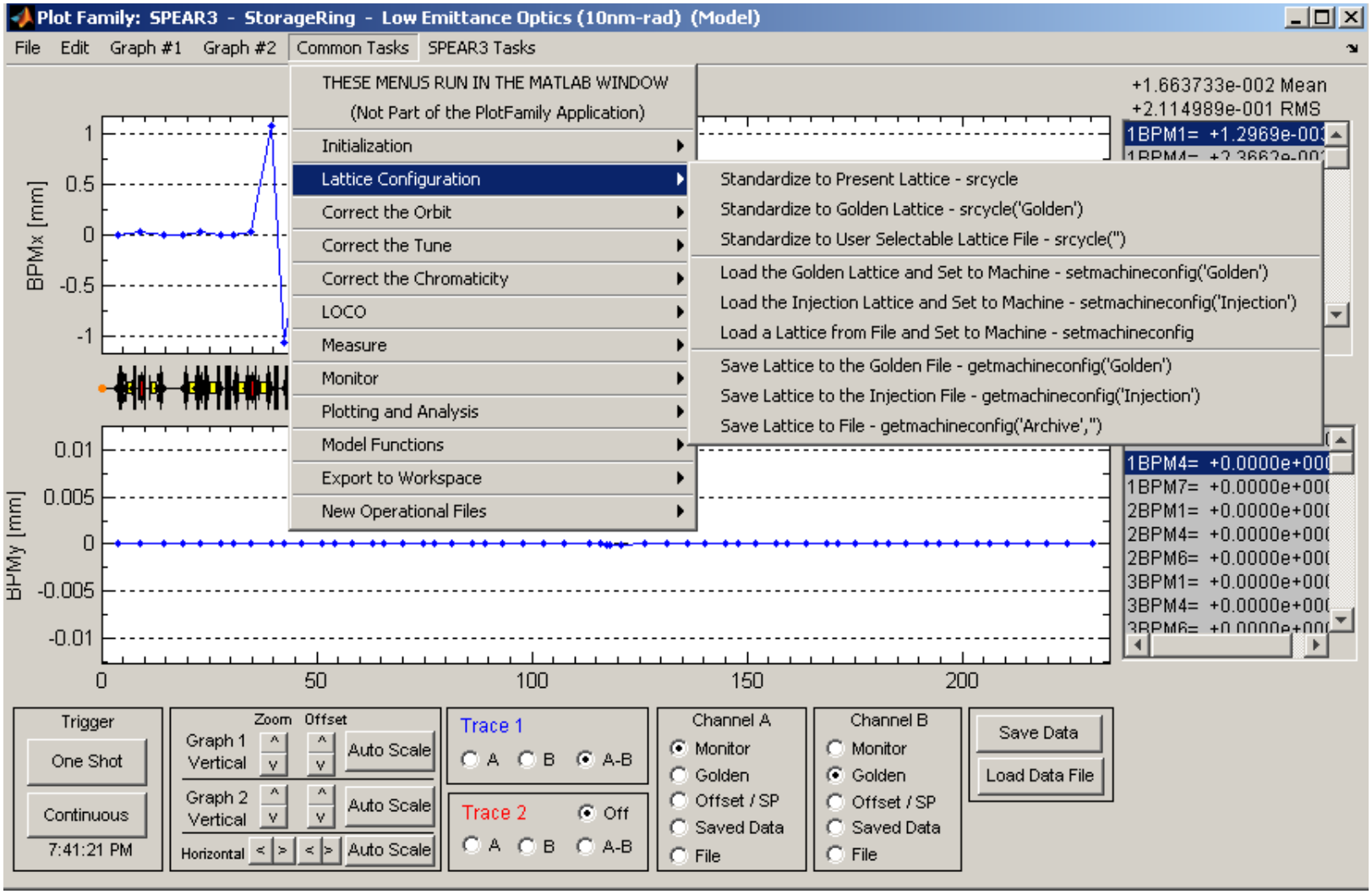
# MiddleLayer Data Flow Diagram



# High Level Applications

- Magnet lattice save / restore / configuration control
- Energy Ramping
- Slow orbit feedback
- Insertion device compensation
- Quadrupole centering
- Display (plotfamily) / Diagnostics
- LOCO (Response matrix analysis)

# plotfamily


Plot Family: SPEAR3 - StorageRing - Low Emittance Optics (10nm-rad) (Model)
File
Edit
Graph #1
Graph #2
Common Tasks
SPEAR3 Tasks
THESE MENUS RUN IN THE MATLAB WINDOW
(Not Part of the PlotFamily Application)
Initialization
Lattice Configuration
Correct the Orbit
Correct the Tune
Correct the Chromaticity
LOCO
Measure
Monitor
Plotting and Analysis
Model Functions
Export to Workspace
New Operational Files
Standardize to Present Lattice - srcycle
Standardize to Golden Lattice - srcycle('Golden')
Standardize to User Selectable Lattice File - srcycle('')
Load the Golden Lattice and Set to Machine - setmachineconfig('Golden')
Load the Injection Lattice and Set to Machine - setmachineconfig('Injection')
Load a Lattice from File and Set to Machine - setmachineconfig
Save Lattice to the Golden File - getmachineconfig('Golden')
Save Lattice to the Injection File - getmachineconfig('Injection')
Save Lattice to File - getmachineconfig('Archive','')
+1.663733e-002 Mean
+2.114989e-001 RMS
1BPM1= +1.2969e-00
1BPM4= +0.0000e+00
1BPM7= +0.0000e+00
2BPM1= +0.0000e+00
2BPM4= +0.0000e+00
2BPM6= +0.0000e+00
3BPM1= +0.0000e+00
3BPM4= +0.0000e+00
BPMx [mm]
BPMy [mm]
Trigger
One Shot
Continuous
7:41:21 PM
Zoom
Offset
Graph 1 Vertical
Graph 2 Vertical
Horizontal
Auto Scale
Trace 1
Trace 2
A
B
A-B
Off
Channel A
Channel B
Monitor
Golden
Offset / SP
Saved Data
File
Save Data
Load Data File

# Scripting Example: Orbit Correction

```
% Create an Orbit Error
vcm = .25 * randn(73,1);     % 73 vertical correctors at the ALS
setsp('VCM', vcm);

% Get the vertical orbit
Y = getam('BPMy');

% Get the Vertical response matrix from the model
Ry = getrespmat('BPMy', 'VCM');       % 122x73 matrix

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

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

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

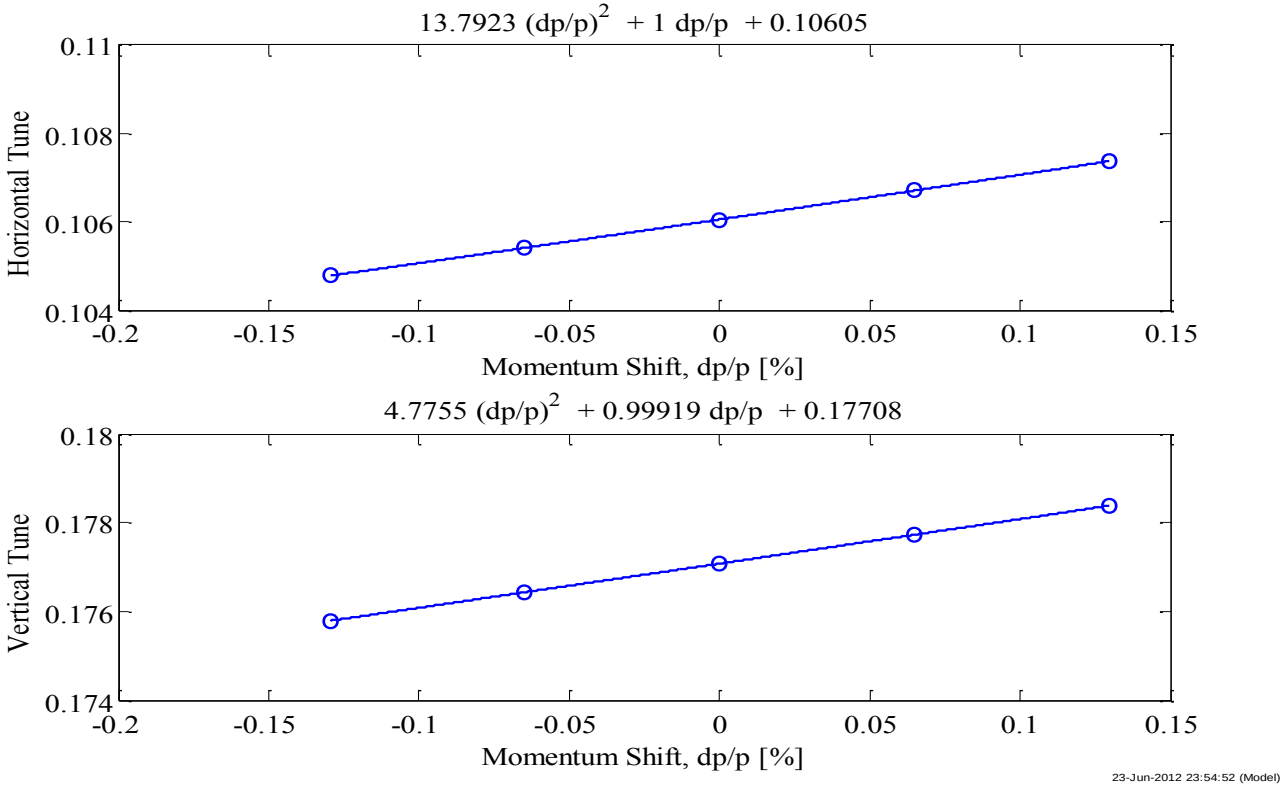
# Tune Measurement and Correction

```
% Measure the tune (just to check the result)
Tune1 = gettune;

% Get the chromaticity response matrix for SF and SD
m = gettuneresp;              % Default (used by settune, steptune)
m = gettuneresp('Model');     % Model

% Compute the delta QF and QD and apply the correction
DeltaAmps = inv(m) * [-.05; .05];
setsp({'QF', 'QD'}, {getsp('QF')+DeltaAmps(1), getsp('QD')+DeltaAmps(2)});

% Measure the chromaticity and check result
Tune2 = gettune;
DeltaTune = Tune2 - Tune1
```

The above tune change can be done with

```
% step the tune
steptune([-.05; .05];);
```

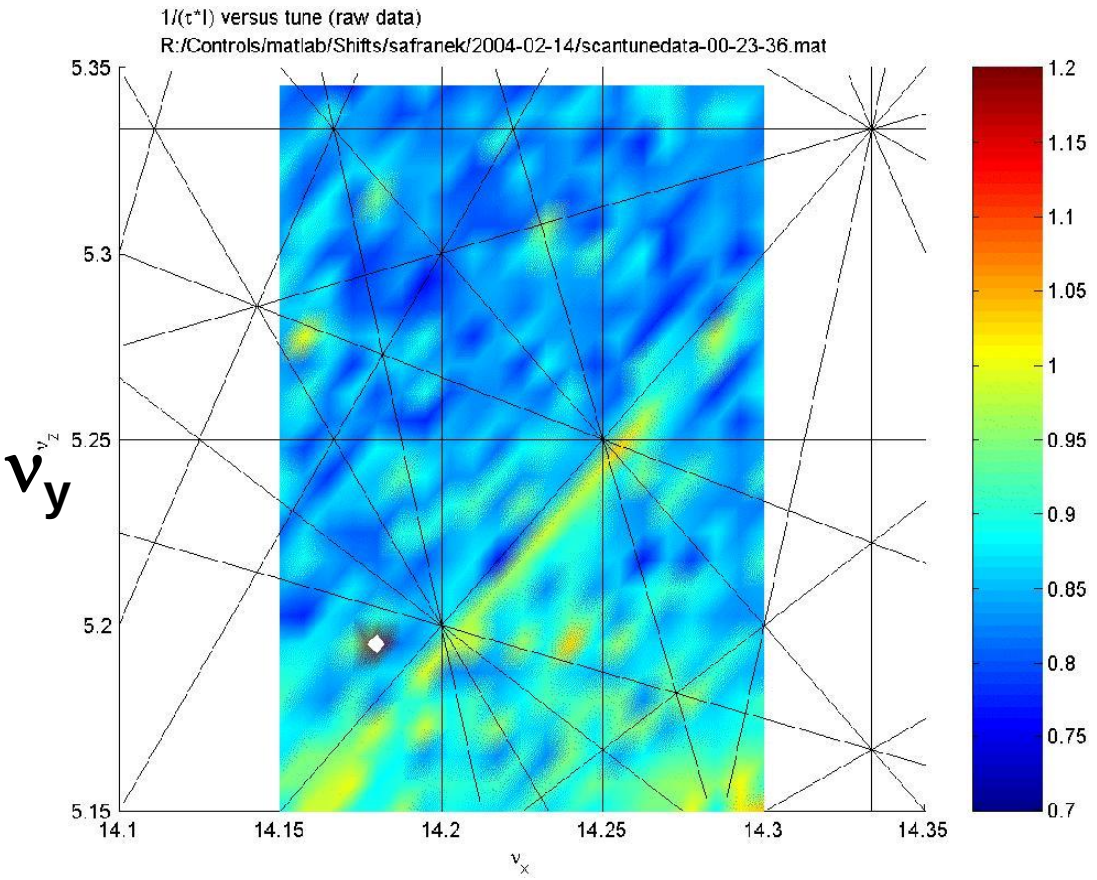
# Chromaticity Measurement and Correction




- measchro
- measchroresp
- plotchro
- setchro
- stepchro
- getchro
- getchroresp

# Lifetime vs. Tunes

- Resonant line:
  - $\nu_x - \nu_y = 9$
- Operating tunes (14.19, 5.23)
- Data gathered automatically on owl shift.



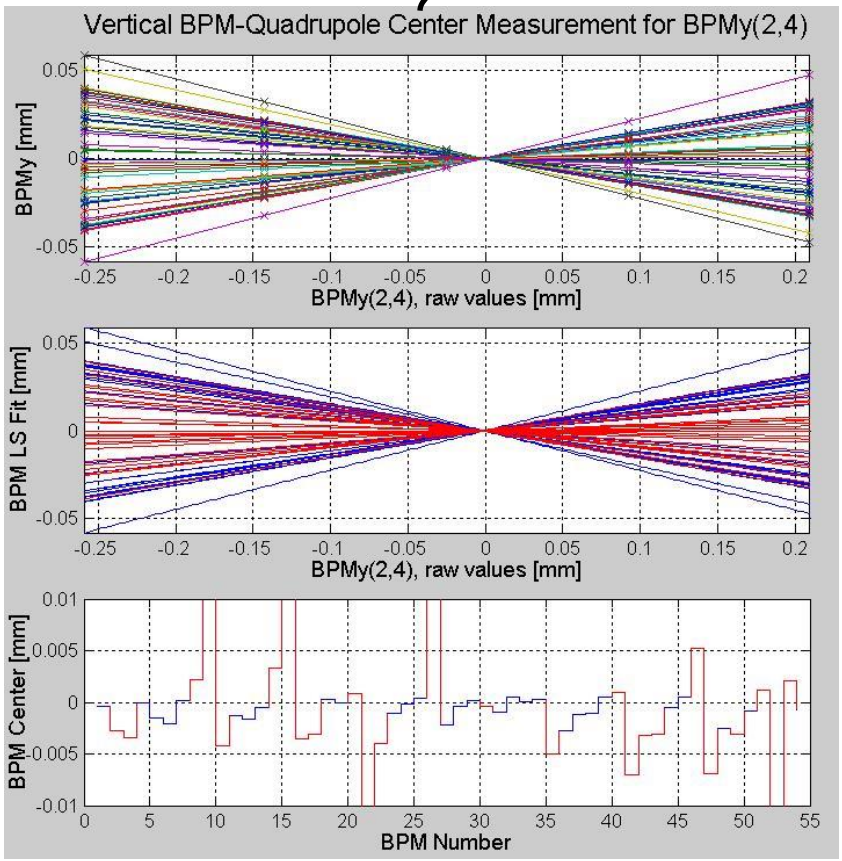

scantune

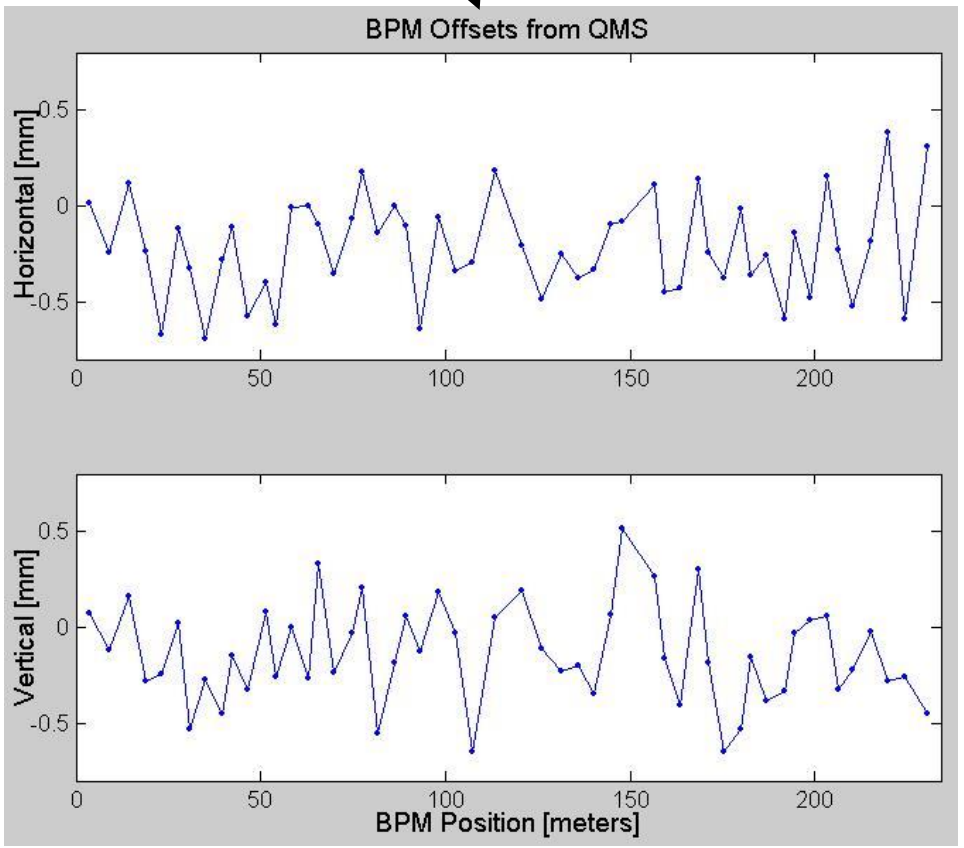
**5.15**

$\nu_y$

$\nu_x$

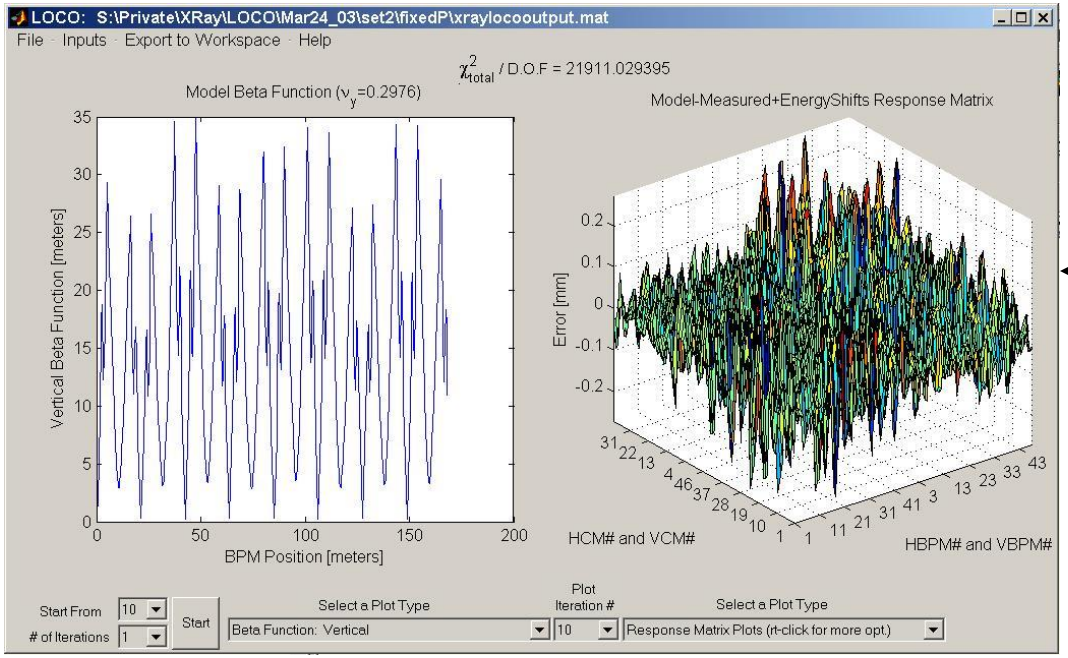
# Beam-based alignment (quadrupole centering)



# LOCO Optics Analysis

- Calibrate/control optics using orbit response matrix
- Determine quadrupole gradients
- Correcte 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

# Outline of computer lab sessions

- Monday
  - Matlab Middle Layer (MML)
  - Orbit correction
  - Beam based alignment
- Tuesday
  - Turn-by-turn BPM data analysis
- Wednesday
  - Optics and coupling correction with LOCO
- Thursday
  - Online optimization with RCDS and other algorithms

Matlab scripts and accompanying instructions are provided for each session. Open the scripts and follow the instructions.

# Summary

- Matlab Middle Layer was created to provide an easy environment for accelerator physicists to interact with the machine.
- It includes
  - A link between the programming environment and the low level controls system
  - Physics subroutines commonly used in operation and machine studies
  - Data management
  - High level applications
  - Simulation environment