USPAS 2015 June – Beam Based Diagnostics

Computer Class – Session 1

Xiaobiao Huang (6/12/15)

Goals: (1) Get familiar with Matlab Middle Layer (MML) and Accelerator Toolbox (AT).
(2) Practice orbit correction.
(3) Beam based alignment (BBA) with MML.

Part 1: Matlab Middle Layer (MML)

Instruction: Run the Matlab script "Monday_MML.m" cell by cell. Follow the instructions for each cell below.

Cell 1: set up MML

>> setpathspear3

This sets up the MML environment for the SPEAR3 storage ring. The accelerator object, accelerator data structure and the machine specific paths are defined. You can set up for other machines, too. For example, for ALS, use

>> setpathals

Cell 2: Inspect the Accelerator Object

>> ao = getao

The returned structure "ao" contains all family data for the machine. Each family is a group of similar devices. You can further inspect the content of the families. For example, run command

>> ao.QF

>> ao.QF.Setpoint

>> ao.QF.Setpoint.ChannelNames

The last command list the corresponding setpoint PV names in the control system for all the QF magnets. Similarly all QF magnet monitor PV names are listed by

>> ao.QF.Monitor.ChannelNames

MML does 'translation' between physicist's convention and the control system with the family data stored in the Accelerator Object. Therefore we need not to memorize or look up the PV names to access these devices.

Cell 3: Check the geometry of the machine

>> intlat

The AT function "inatlat" provides a GUI to help examine the machine geometry and explore the accelerator elements. You can zoom in and click on an element to see its properties.

Cell 4: Plotfamily

>> plotfamily

“plotfamily” is a summary GUI for many convenient tools provided by MML. By default it displays the beam orbit. Click the button “One Shot” or “Continuous”. Click “Auto Scale” to change vertical scale if necessary. You can choose to display other data such as magnet strengths using the menu.

Cell 5: Access the control system: read setpoint or monitor

You can access a whole family of devices at once or one or a few selected devices. Example,

```
>> getsp(‘QF’)
>> getsp(‘QF’,[3, 1])
```

The setpoint of QF [3,1] can also be read directly with the channel PV, using (works only in the online mode)

```
>> getsp('03G-QF1:CurrSetpt')
```

But this form is not convenient since one needs to remember the PV name.

Cell 6: Access the control system: change setpoint

You can set or step the setpoints with family names and device numbers. Example,

```
>> stepsp('QF',1.0,[3,1]);
```

steps QF [3, 1] by 1.0 Amp, while

```
>> setsp('QF',71.9,[3,1]);
```

sets the setpoint of QF [3, 1] to 71.9 Amp.

To check the changes to the model, one can calculate the betatron tunes (using MML function “gettune”) before and after the quadrupole setpoint is changed.

Cell 7: calculate optics functions

It is often necessary to know the optics functions at certain locations of the storage ring. This can be easily done in MML using functions like “modelbeta”, “modeldisp”, and “modeltune”. Example,

```
>> [betx,bety] =  modelbeta('BPMx');  %at all BPMs
>> [betx1,bety1] =  modelbeta('QF', [3,1]);  %at QF [3, 1]
>> [Dx, Dy] = modeldisp('BPMx');
```

Cell 8: Examine an AT lattice model file and source code for an AT passmethod

```
>> edit sp3v82
```

This opens the function “sp3v82.m” which defines a lattice for SPEAR3 for AT. In the Matlab Editor you can examine how the lattice is defined.

AT does particle tracking using “passmethods” for various types of accelerator elements. The passmethods are usually written in C and are compiled to Matlab callable binary, mex file (Matlab executable). An example of the source code can be seen with

```
 >> edit QuadLinearPass.c
```

where “QuadLinearPass” is a passmethod for quadrupoles that is based on linear transfer matrix.

Part 2: Orbit Correction

Open the script “monday_Orbit.m” and follow the instructions below to execute the commands in the cells.

Cell 1: Load a SPEAR3 model and view the orbit

Load the SPEAR3 low emittance lattice

```
>> sp3v82_lelt
```

The orbit can be viewed with “plotfamily”. So open “plotfamily” if you have closed it.

```
>> plotfamily
```

Click “One Shot” to update the orbit.

Cell 2: Step the RF frequency and observe orbit changes

```
>> drf = 0.001;
>> steprf(drf)
```

Step the RF frequency by 1 kHz. The pattern in horizontal orbit change is proportional to the dispersion function. After viewing the orbit change, press “Enter” in the command window to execute the next command of setting RF frequency back to original.

```
>> setrf(rf0)
```

Update “plotfamily” again.

Cell 3&4: Orbit variation due to radiation energy loss

For an electron storage ring, the beam loses energy at bending magnets and gains energy at RF cavities. When all correctors are off, this energy variation is reflected on the horizontal orbit due to dispersion. We can see that by turning on radiation and the cavity for the model.

```
>> setcavity on
>> setradiation on
```

Now update the orbit with “plotfamily”. In real operation this small variation is corrected along with other sources of orbit errors. With the bare lattice of SPEAR3 the beam loses 0.88 MeV in one turn.

Cell 5: Introduce orbit errors with one corrector (for each plane)

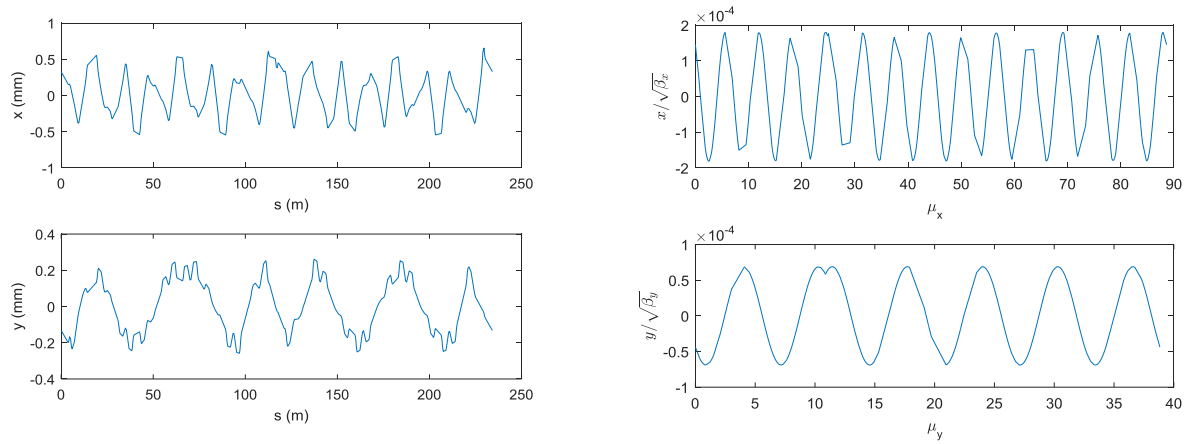
Change the setpoint for one horizontal corrector magnet (HCM) and one vertical corrector magnet (VCM) to cause orbit errors. View the orbit in “plotfamily”.

```
>> setsp('HCM', 1, [6 1]) ;
>> setsp('VCM', 1, [6 1]);
```

Count the peaks of the perturbed orbits. The betatron tunes of this lattice are [14.106, 6.177].

Cell 6: Inspect the closed orbit with normalized coordinate

Plot the closed orbit vs. s-position and the normalized orbit coordinate vs. betatron phase.



The closed orbits in $\frac{x}{\sqrt{\beta_x}}$ and $\frac{y}{\sqrt{\beta_y}}$ coordinates are smooth sinusoidal functions of the betatron phases, with kinks at where kicks are applied.

Cell 7: Correct orbit with MML

The MML function "setorbitdefault" correct the orbit to default target as defined by BPM "offset" values. Correct the orbit and then check the strengths of the two corrector magnets.

```
>> setorbitdefault
>> getam('HCM',[6,1])
>> getam('VCM',[6,1])
```

After correction, the correctors are set back to nearly zero.

Cell 8: Reset orbit by setting all corrector magnets to zero strength

```
>> setsp('HCM',0);
>> setsp('VCM',0);
```

We clear the orbit and prepare for the next test.

Cell 9: Introduce orbit errors with quadrupole misalignment at one quad

Misalignment in AT model can be introduced by shifting the particle coordinate at the entrance and exit faces of a magnet to opposite directions (valid for magnets that are not too long).

We shift QF [3, 2] by 0.1 mm both horizontally and vertically, which causes orbit errors.

```
>> qfi = family2atindex('QF', [3,2]);
>> DeltaX = 0.1e-3;
>> DeltaY = 0.1e-3;
>> THERING{qfi}.T1(1) = -DeltaX;
>> THERING{qfi}.T2(1) = DeltaX;
>> THERING{qfi}.T1(3) = -DeltaY;
>> THERING{qfi}.T2(3) = DeltaY;
```

Here we operate directly on the lattice model (THERING). Check the orbit in "plotfamily" after the changes. A small quadrupole alignment error can cause large orbit errors – an amplification effect.

Cell 10: Orbit errors by misalignment at all quads

We now introduce random misalignment errors to all 99 quads of SPEAR3 and check the orbit. The alignment error sigmas are assumed 0.1 mm for both planes. After executing this cell, update the orbit with "plotfamily".

Cell 11: Correct the orbit errors

Again correct orbit with

>> setorbitdefault

Do this multiple times. Each time check the orbit.

Note: orbit correction can also be done with "plotfamily". Use menu item: Common Tasks → Correct the Orbit. Choose one of the options. This will give you more control of the correction calculation, such as the number of singular values to be used.

Cell 12: Clear the orbit errors

Clear all quadrupole misalignments and corrector values.

Cell 13: Create a closed orbit bump

It is often necessary to create local orbit bumps in storage ring operation. The MML function "setorbitbump" is a convenient tool for that. One needs to supply the BPMs, the desired orbit changes, and the correctors to be used.

```
>> DetlaX = -1;
>> [OCS] = setorbitbump('BPMx', [3  6; 4, 1 ], [1 1]*DetlaX , 'HCM', [ -4 -3 -2 -1 1 2 3 4 ], 1, [], 'NoSetSP');
>> stepsp( OCS.CM.FamilyName, OCS.CM.Delta, OCS.CM.DeviceList );
```

This creates a -1 mm horizontal orbit bump at the SPEAR3 septum magnet.

Cell 14: Get closed orbit directly with AT

In the simulator mode, the closed orbit is calculated with AT using the lattice model. The closed orbit can be calculated directly with AT (without using MML functions). The functions to use are "findorbit4" or "findorbit6". The latter requires an RF cavity.

Cell 15: Introduce vertical orbit errors

Use a random set of VCM values to generate vertical orbit errors.

Cell 16: Correct orbit with your own code

Correct the orbit errors by inverting the response matrix. The vertical response matrix is 57×56 in dimension. The number of singular values to keep is 55 in this test. You can try to change it.

Part 3: Beam based alignment (BBA)

With BBA, we find the centers of quadrupoles. If we steer the beam through the center of a quadrupole magnet, a change of its strength will not cause orbit shift (i.e., no 'feed-down' effect).

Cell 1: Initialization

Reload the lattice. Launch "plotfamily" if it is not up.

Cell 2: Plant in offset errors to a quadrupole

We put in a 0.12 mm horizontal and -0.08 mm vertical offset to the center of quadrupole QF [7, 1]. Check the orbit with "plotfamily".

Cell 3: Correct the orbit

>> setorbitdefault

Correct orbit and check it with "plotfamily".

Cell 4: Run QMS for the quadrupole

>> quadcenter('QF',[7,1],0)

This launches the MML routine for BBA for the quadrupole. The position of the center is measured with the nearest BPM, in this case, BPM [7, 1].

Cell 5: Check QMS results

To access previous QMS data, use

>> quadplot

and open the data file. Data will be plotted in figures and printed to the command window.

Cell 6: Apply offset to BPM

The BPM offset (orbit offset at a BPM when the beam passes through the center of a nearby quadrupole) data are stored in a PhysData file, which can be accessed with "getphysdata" and "setphysdata". Use these functions to read the original offset values for BPM [7, 1] and set the new values found with QMS. Also change the BPM golden values (orbit target).

Cell 7: Correct orbit again

After BPM offset is changed, correct orbit again. You won't see much change on "plotfamily" if displaying the orbit in "A-B" mode with B the BPM offset or golden. Choose to display BPM monitor data (A for "Trace 1").

Cell 8: Clean up

Restore the lattice and PhysData.