USPAS 2015 June – Beam Based Diagnostics

Computer Class – Session 2

Xiaobiao Huang (6/12/15)

Goals: Practice turn-by-turn BPM data analysis with various techniques.

Part 1: Phase measurement with Castro's method

Assume $x_i = \sin(2\pi i\nu + \mu)$, for $i = 1, 2, \cdots, N$, then the phase $\mu = \mathrm{atan}\frac{C}{S}$, with $C = \sum_i x_i \cos 2\pi i\nu$ and $S = \sum_i x_i \sin 2\pi i\nu$, where ν is the tune (P. Castro, et al, PAC 93).

Open the script "tuesday_phase.m" and follow the instructions.
(This part of the program was modified from an earlier version by Christoph Steier, et al).

Cell 1: Initialization
Set MML path for SPEAR3.

Cell 2: Plot the beta functions for the nominal lattice
Use MML function "modeltwiss" to obtain beta functions and phase advances and plot beta (β) vs. phase (μ).

Cell 3: Add lattice errors by changing one quadrupole
Increase QF [6, 1] by 1 Amp and re-calculate the beta and phase. Plot the comparison to the nominal lattice and the relative beta changes.

Put QF [6, 1] to original value when calculation is done.

Cell 4: Simulate turn-by-turn BPM data
Track one particle for 1024 turns, with initial x and y offsets of 0.1 mm. Record the orbit readings at all 57 BPMs. No noise is added at this time. Data is saved to a file "data_noqerr_nonoise".

Cell 5: Calculate the tunes and phases from the simulated TbT BPM data
First the tunes are determined from data recorded at each BPM with function "findfreq.m". FFT with sine window and interpolation are used to determine the tune. Then the betatron phase advances at each BPM is determined with function "calcphase.m" using P. Castro's approach.

The phase advances "measured" from the simulated TbT BPM data are compared to the nominal values and the difference is plotted for both transverse planes. The difference is very small.

Cell 6: Add random noise to BPM data and re-calculate phases

Random Gaussian noise with a sigma of 10 micron is added to all simulated BPM readings. Tune and phase measurements are repeated for the same simulated data (now with noise).

Differences of “measured” phases (with noise) from the nominal phases are added to the plot (Figure 102). Noise introduces significant errors to the phase measurement.

Cell 7: Change one quadrupole
Increase QF [6, 1] by 1 Amp and re-calculate beta functions and phases. Plot the actual phase changes due to the quadrupole step change as a reference (Figure 103).

Cell 8: Simulate TbT BPM data for the lattice with quadrupole error
With the quadrupole error, re-do the TbT data tracking and record data at all 57 BPMs. Add Gaussian noise with a sigma of 10 micron to all BPMs.

Calculate the “measured” phase advances again with “findfreq.m” and “calcphase.m”. Add the phases to the plot (Figure 103) for comparison.

Step down QF [6, 1] by 1 Amp (back to original value).

Part 2: Precise tune determination

We evaluate the precision of tune determination for several methods:
NAFF (Laskar et al): maximize the overlap of power spectrum of the data and the sinusoidal signal.
IPFA (Bartolini et al): interpolated FFT.
Findfreq (Steier): FFT with sine window and interpolation.

Open the script “tuesday_naff.m” and execute the cells as explained below.

Cell 1: 128 turns, no noise
With tune $\nu = \pi - 3 \approx 0.1415926$, generate a discrete sine function data set for 128 turns with unit amplitude (1 mm). No noise is added. Evaluate the tune from this data sets with the three methods. Errors of the tune determination are shown in the order of NAFF, Findfreq and IPFA.

Cell 2: 128 turns, w/ noise
Add Gaussian noise of 20 microns to the data. Re-evaluate the tunes and print out errors.

Cell 3: 512 turns, no noise
With the same tune, generate a data set of 512 turns with amplitude of 1 mm. Evaluate tune determination errors.

Cell 4: 512 turns, w/ noise
Add Gaussian noise of 20 microns to the data. Re-evaluate the tunes and print out errors.
All three methods have good performance.

Part 3: Extracting frequency components

The precise tune determination methods can be used to extract the frequency components of measured signals. In the following we use NAFF.

Open the script “tuesday_fc.m” and execute the cells.

Cell 1: Initialization

Load SPEAR3 lattice and insert skew quadrupole errors for two magnets.

Cell 2: Tracking for TbT BPM data

Track 1050 turns and record positions at all BPMs. The initial orbit offsets are 2 mm horizontal and 1 mm vertical, respectively.

Cell 3: Plot the Fourier spectrum for one BPM

Plot the FFT amplitude (1024 turns) in log-scale for both X and Y planes for one BPM. Linear and nonlinear coupling components can be seen, such as $\nu_y - \nu_x$, $\nu_y + \nu_x$, $2\nu_x$, $2\nu_y$.

Cell 4: Remove the leading component

Remove the leading component with NAFF. Plot the Fourier spectrum of the original signal along with the residual signal (with the leading component removed).

Cell 5: Iteratively remove the frequency components

Remove the 10 leading component iteratively. Plot the Fourier spectrum of the original and residual signals. List the tunes and strengths of the frequency components. The strengths and phases of the frequency components at all BPMs can be used for optics and coupling correction and nonlinear dynamics studies.

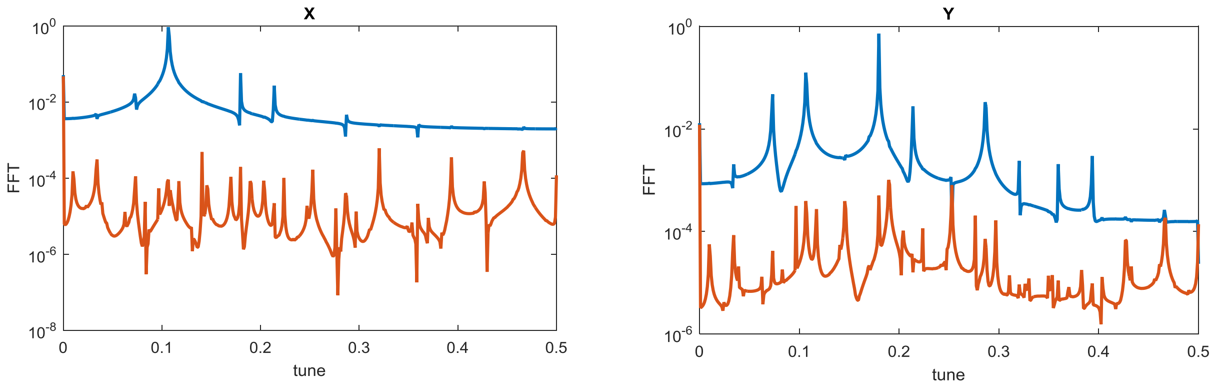

Figure 1: the Fourier spectrum of tracking data for one BPM before and after the 10 leading components are extracted and removed. SPEAR3 lattice with coupling is used for tracking. No noise is added to data.

Repeat the above from Cell 3 with random noised added to the BPM signals (uncomment line 56-57). Many of the oscillations will be swamped by the noise.

Part 4: Independent component analysis (ICA)

We will learn to use the ICA to analyze TbT BPM data. By separating the frequency component consistently for all BPMs, we can derive beta functions and phase advances for the normal modes (decoupled betatron motion) with high accuracy. Amplitudes and phases of the coupled linear and nonlinear components can also be obtained.

Cell 1: Explore the ICA Matlab GUI
Launch the Matlab GUI of ICA with

```
>> icagui
```

On the far right, use the two top boxes to select data sets (the first one for a data category, the second one for a data set). The data options can be changed by editing the beginning of “icagui.m” and correspondingly the script “icaloaddata.m”.

After selecting the data set, click “Run” to execute the analysis. With the noise level set to zero (default value), a popup window will show the singular value spectrum and one can select the number of modes to keep. You can set the “noise level” window to a negative integer to set the number of singular values. For example, setting the value to -16 will select 16 SVD modes for analysis.

The four plots show the diagonal values of the diagonalized time-delayed convariance matrix (top left), the temporal pattern (top right), the spatial pattern (bottom left), and the Fourier spectrum of the temporal pattern. Click the “>>” and “<<” buttons to explore the modes.

The default analysis method is “ICA”. This can be changed to PCA (i.e., SVD).

Cell 2: Use the command line ICA function
The Matlab ICA command line function is “icacmd.m”. One can provide the GUI parameters in a data structure and pass this function and perform the ICA data analysis.

```
>> dout = icacmd('run', [BPMx; BPMy],paraset);
```

The “A” and “s” fields of variable “dout” contain the spatial and temporal patterns of the ICA modes, respectively. The modes can be automatically paired up, with

```
>> [dummy,pairs] = icacmd('pair', dout);
>> pairs
```

One can check the paired modes by viewing their Fourier spectrum.

```
>> plotfft(dout.s(pairs(1,:),:))
```

Cell 3: Calculate betatron phases with ICA modes – data from SPEAR3 nominal lattice
Using ICA to analyze the saved tracking TbT BPM data for the SPEAR3 nominal lattice (saved while we were working on “tuesday_phase.m”), we derive the beta functions and phase advances from the spatial patterns of the ICA modes. This analysis is done in “icatest.m”. The results are plotted along with the results of Castro’s method. The error for the ICA method is much lower than Castro’s method.

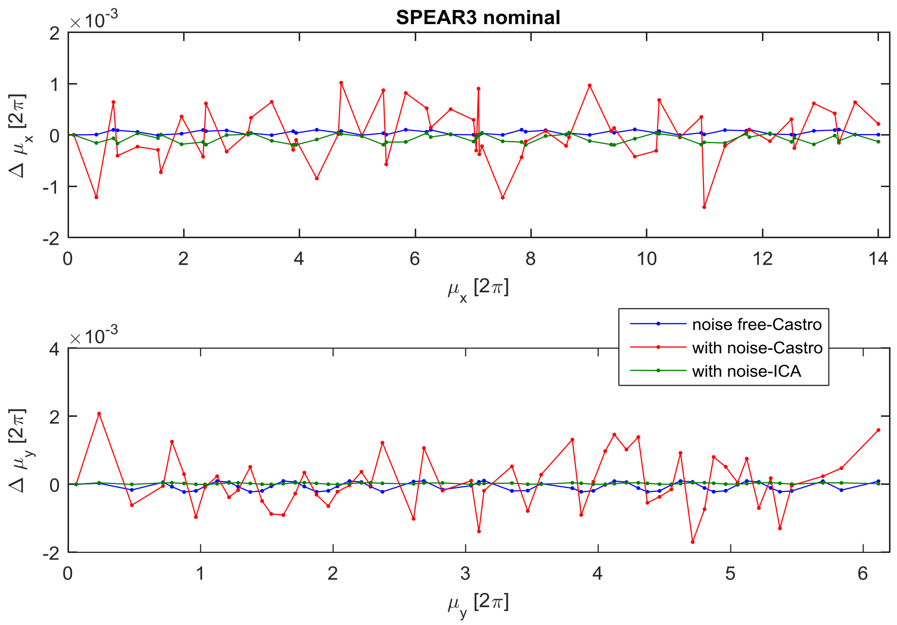

Figure 2: the errors of phase advances obtained from tracking data for the SPEAR3 nominal lattice for three cases: (1) no noise in data, w/ Castro's method for phase measurement: (2) with noise in data, Castro's method; (3) with noise in data, using ICA.

Cell 4: Calculate betatron phases with ICA modes – data from SPEAR3 lattice w/ QF [6, 1] error
The same analysis for the tracking data with quadrupole QF [6, 1] increased by 1 Amp from nominal value. Zoom in to see that the ICA curve overlaps with the target curve.

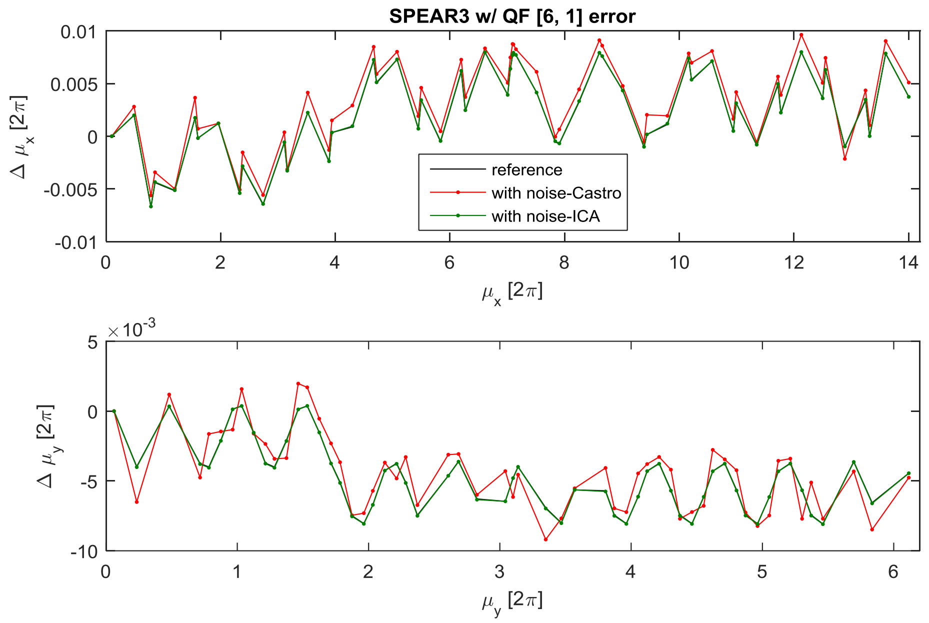

Figure 3: Phase advance errors introduced by QF [6, 1], w/ +1 Amp current increase. The ICA results overlaps with the actual reference value, while Castro's method shows big errors.

Part 5: Decoherence in TbT BPM data

Open and execute the script “tuesday_decoherence.m” to study the decoherence effect on TbT BPM measurements. BPM sees the average position of the bunch. When the bunch of particles is spread out in the phase space (decoherence), the measured oscillation amplitude does not reflect those of the individual particles in the bunch. Nonlinear detuning at large oscillation amplitude and large chromaticity can cause decoherence.

We track a bunch of particles with the nominal SPEAR3 longitudinal distribution.

Cell 1: Nominal chromaticity

At the nominal chromaticities [2, 2].

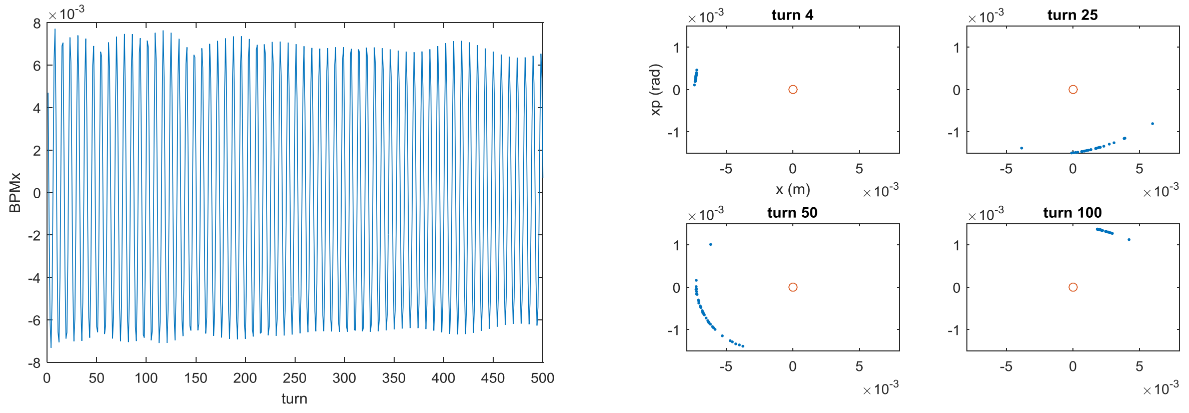

Figure 4: Horizontal position (left) and (x, x') phase space (right) with chromaticity at [2, 2].

Cell 2: Increased chromaticity

Increase chromaticities to [6, 6].

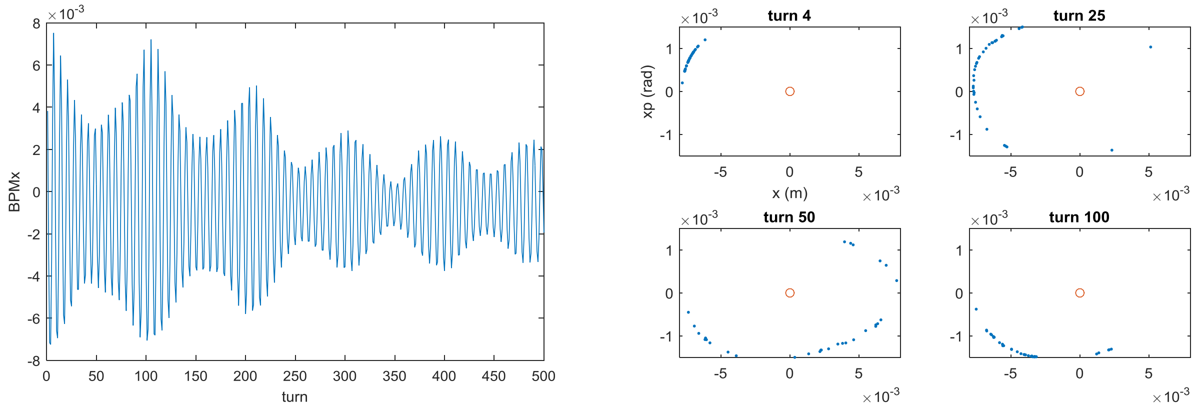

Figure 5: With chromaticities at [6, 6].