Multi bunch feedback application software

Alun Morgan

April 2024

Contents

1 Conceptual descriptions **2**
- 1.1 Overview . . . 2
  - 1.1.1 Frontend Optimisation . . . 3
  - 1.1.2 Measurements . . . 3

2 Practical use **5**
- 2.1 Code location . . . 5
- 2.2 MBF frontend operational setup . . . 5
  - 2.2.1 DORIS phase . . . 5
  - 2.2.2 Machine settings . . . 7
  - 2.2.3 Frontend phases . . . 7
- 2.3 MBF closed loop setup . . . 10
  - 2.3.1 Machine settings . . . 10
  - 2.3.2 Initial measurement . . . 10
  - 2.3.3 Assessment . . . 11
  - 2.3.4 Tuning the loop delay . . . 12
- 2.4 MBF standard operational measurements . . . 14
  - 2.4.1 Machine setup . . . 14
  - 2.4.2 Code to run . . . 14
- 2.5 MBF developmental measurements . . . 15
  - 2.5.1 Transverse damping . . . 15
  - 2.5.2 Emittance control . . . 15
  - 2.5.3 Bunch motion . . . 15

3 MBF Archival retrieval **16**
- 3.1 Introduction . . . 16
- 3.2 Getting the latest data . . . 16
- 3.3 Frontend tuning results . . . 20
- 3.4 Getting any historic data . . . 22

Chapter 1

Conceptual descriptions

1.1 Overview

The hardware for multi bunch feedback has many different possible configurations. Broadly speaking a feedback system has a frontend which takes in the signals from the pickups, and does the initial analogue signal conditioning. These signals are then fed into digital systems for each of the three planes (x,y,s). These digital systems calculate the feedback required and send signals to analogue drive electronics in order to excite the beam in the given plane. This is summarised in figure 1.1

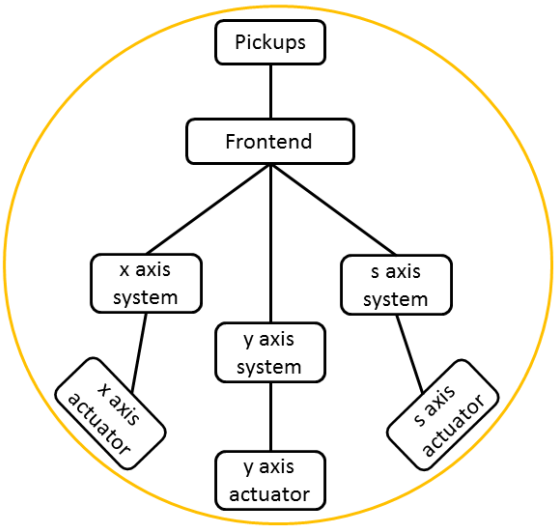


Figure 1.1: System overview

In order to make such a system more easily usable for a wide range of system users, this suite of task specific applications has been developed. The aim of the suite is to abstract away the implementation detail of the hardware and thus allow the users to put their attention on type of measurement they want to do. This abstraction also aids in the development of bespoke measurements for specific investigations.

All these codes save the relevant data and graphs to a file structure based on the date, and given unique names based on the date and time the application was run. The root path for this structure is defined in `mbf_system_config.m`, as is the harmonic number of the machine under study. In general all machine/deployment parameters are set in this file. The rest of the codebase

it written to adapt to what is set here.

In addition to the system data from the multi bunch hardware, more general machine parameters are also captured for growdamp, modescan and bunch motion codes, in order to enable machine studies (e.g current dependence). This additional data is set in `machine_environment.m`.

1.1.1 Frontend Optimisation

In order to work optimally the feedback systems need to be matched in timing/phase with the beam phase (not the RF phase) and be tuned so that the sampling point coincides with the maximum signal from each bunch.

The first requirement is done by phase shifting the input signal. At its most basic this comprises of a single phase shifter for each channel which both allows each channel to be adjusted to follow changes in beam phase, and to allow the axis to be tuned in order to account for differences in the phase delay of that channel.

The practical implementation will depend on the details of your frontend hardware. However is often realised as a single LO phase shifter which is common to all axes and handles the beam phase adjustments. meanwhile each channel has an individual additional phase shifter to tune out the systemic differences in phase delay between the channels. Such a setup is shown in figure 1.2 Another variation is replacing the LO phase shifter with an automatic beam phase following system.

The second requirement is done by adjusting the sampling clock until a maximum signal is obtained with a minimum signal leakage to adjacent RF buckets.

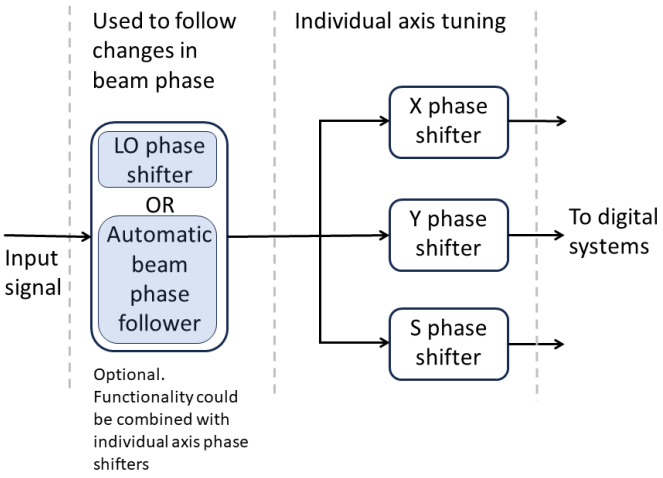


Figure 1.2: phase shifters

Note: The Libera bunch by bunch frontend seems to get into problems when scanning the LO phase shifter. It is probably best not to scan the LO phase if you have that frontend.

1.1.2 Measurements

Frontend scans

Scan the relevant phase shifter or clock delay while monitoring the measure signal amplitude. The resulting graphs will allow appropriate working points to be set.

Mode scans

A mode scan is a sequential excitation of the harmonics of the tune frequency. This is mainly a system check in order to identify any performance limitations or degradation.

Spectra

This code captures the requested number of turns from the requested axis and returns the FFT of the centroid motion of each bunch. This is useful for identifying tunes, tune leakage from one axis to another, also for classifying instabilities.

Growdamp

The measurement comprises a series of excitations at set harmonics of the tune frequency followed by capture of the evolution of bunch centroid positions, both with and without feedback active.

Transverse damping

Emittance control

This uses one of the oscillators to excite the selected bunches at one of the sideband frequencies. This has the effect of blowing up the beam.

Bunch motion capture

The three systems are setup to be ready to capture on the next external trigger. Once that trigger arrives, the three systems simultaneously capture bunch centroid motion. This is useful for capturing injection transients.

Chapter 2

Practical use

2.1 Code location

The code is released under an MIT licence. In order to use it please grab it from the code repository or use the following command on the system command line in order to copy it to your current location:

```
git clone https://github.com/alunmorgan/Multi-bunch-feedback-applications.git
```

and add the resulting folder to your Matlab path.

This and related documentation can be found in the documentation folder in the repository

2.2 MBF frontend operational setup

2.2.1 DORIS phase

Upstream of the MBF frontend there is a DORIS unit which locks the MBF to a fixed relationship to beam phase (rather than RF phase) DORIS will lock if there is more than 3 mA in the storage ring.

Before running the code to scan the doris phase, inject 10 mA into a 900 bunch fill pattern. The following code will scan the phase of the DORIS unit to identify an optimum value.

```
>> single_bunch_location = 400
>> DORIS_target_phase_scan(single_bunch_location)
```

Figure 2.1 shows an example of the output.

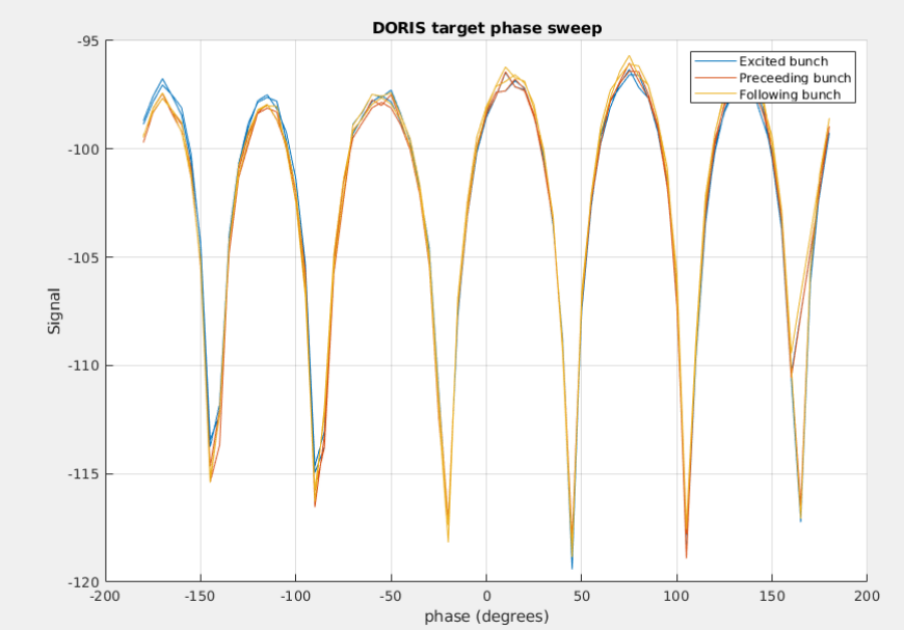


Figure 2.1: In this case 75° looks to be a reasonable working point.

2.2.2 Machine settings

- Inject a single bunch into a selected RF bucket.
- Run the "Zero Bucket" code to ensure MBF bucket numbers are in agreement with the fill pattern from the fill pattern monitor.
- Correct tune values to nominal.

2.2.3 Frontend phases

In the front end there are two phases (delays) we are interested in optimising. The system phase which determines how accurately our detectors are aligned with the target bunch, and the clock phase which determines how close to the peak of the signal we are sampling.

System phase

The following Matlab code will run a system phase scan on each axis sequentially. The graphs will be displayed, and the data is automatically stored.

```
>> single_bunch_location = 400
>> BBBFE_system_phase_scan('X', single_bunch_location)
>> BBBFE_system_phase_scan('Y', single_bunch_location)
>> BBBFE_system_phase_scan('S', single_bunch_location)
```

Figure 2.2 shows an example of the output. You are looking for the phase value with the largest gap between the excited bunch and each of the other bunches.

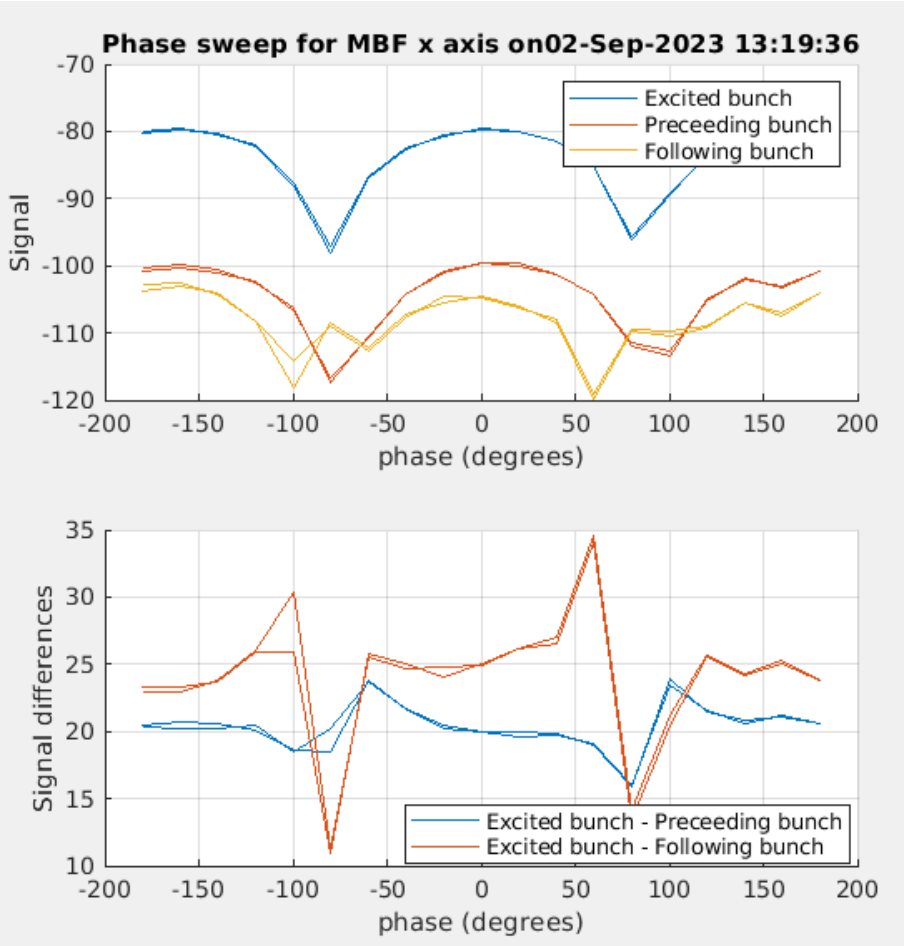


Figure 2.2: In this case $-40°$ looks to be a reasonable working point.

Clock phase

The following Matlab code will run a clock phase scan on each axis sequentially. The graphs will be displayed, and the data is automatically stored.

```
>> single_bunch_location = 400
>> BBBFE_clock_phase_scan('X', single_bunch_location)
>> BBBFE_clock_phase_scan('Y', single_bunch_location)
>> BBBFE_clock_phase_scan('S', single_bunch_location)
```

Figure 2.3 shows an example of the output. You are looking for the phase value with the largest gap between the excited bunch and each of the other bunches.

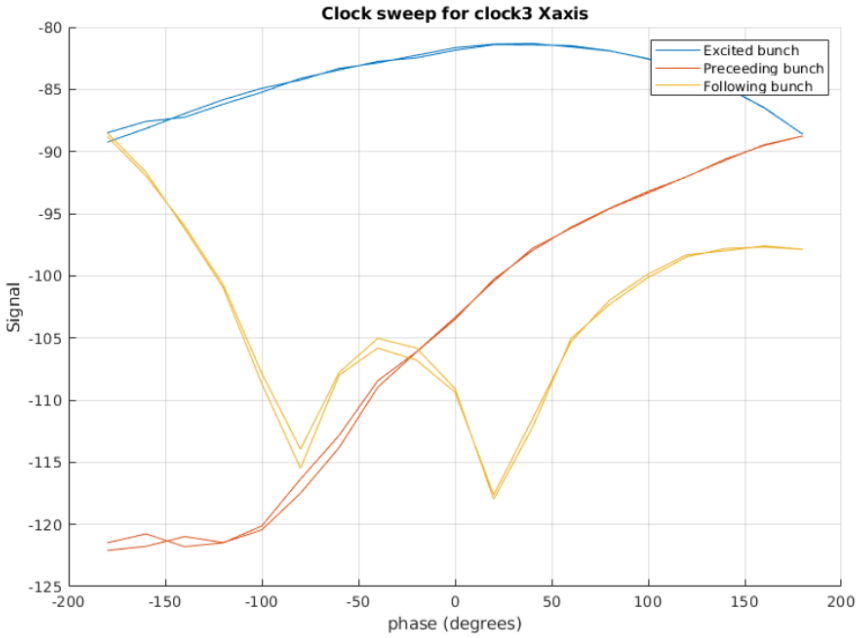


Figure 2.3: In this case $-80°$ looks to be a reasonable working point.

Once these two phases are optimised then the tuning of the front end is complete. The next step is to close the feedback loop.

2.3 MBF closed loop setup

2.3.1 Machine settings

- Inject to 30mA 900 bunches (below the instability threshold)
- Correct tune values to nominal.

2.3.2 Initial measurement

The following Matlab code will run a scan across all the harmonics of the tune on each axis sequentially. The graphs will be displayed, and the data is automatically stored.

```
>> modscan_all('x')
>> modscan_all('y')
>> modscan_all('s')
```

2.3.3 Assessment

Use the modescan results to assess whether the closed loop delay is good enough.

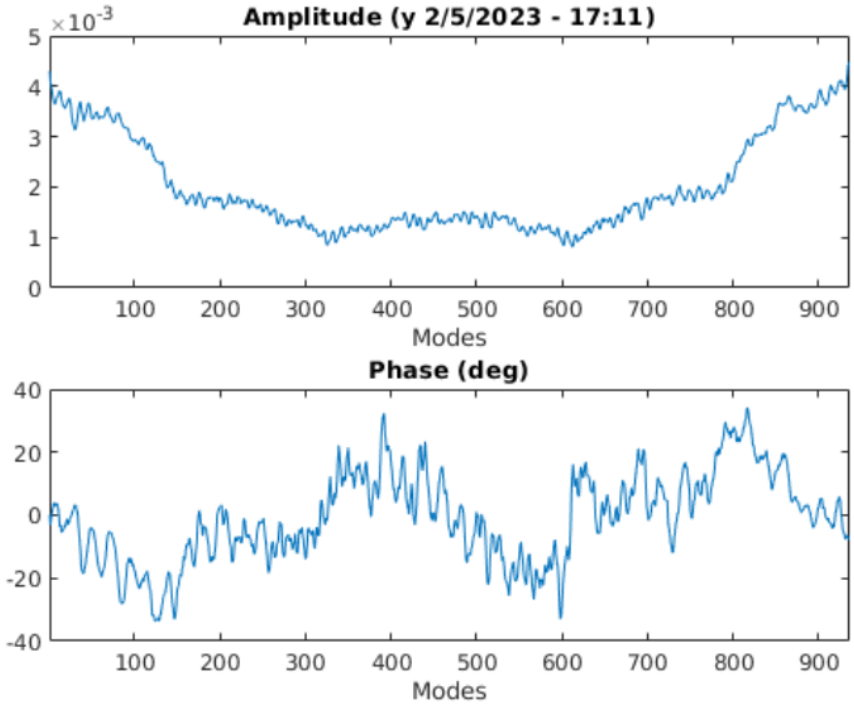


Figure 2.4: In this case the phase is ranging from $-40°$ to $40°$ and there is almost no residual gradient along the modes, which is reasonable. Although any further increase in this range would be concerning and would warrant further investigation.

The amplitude should be fairly symmetric in the modes axis

The phase variation should be much less that $\pm 90°$ and the extents should be balanced between going above and below zero

A large phase variation even when the overall gradient is close to zero indicates that a component is failing – probably the output amplifier. So, a hardware change and a retuning of the frontend is required. (see documentation on tuning up the frontend)

A significant asymmetry in the distribution of the extents of the phase trace (a large gradient in the phase plot) indicates that the overall loop delay is not correct. This can be fixed in the following ways.

First see if a DAC delay can fix it (using the control panel shown in figure 2.5). This is only available in 2ns steps.

Otherwise, we must change cable lengths. The easiest place to do this is at the output port of the output amplifier. As shown in figure 2.6

2.3.4 Tuning the loop delay

Use the `mbf_modescan_setup` Matlab code to setup the measurement and look at the 'tune' screen.

This gives a live modescan you can use for monitoring the impact of your changes. Alternatively you can just rerun the modescan code after each change.

The following Matlab code will set up this visualisation for each axis

Note: You generally work on a single axis at a time

```
>> mbf_modescan_setup('x')
>> mbf_modescan_setup('y')
>> mbf_modescan_setup('s')
```

The aim of this exercise is to get the extents of the phase trace comfortably fit within the ±90° limits (similar to the result show in figure 2.4).

Tuning the DAC delay

This allows additional phase (delay) to be added to the closed loop. However, it can only add in steps of 2ns.

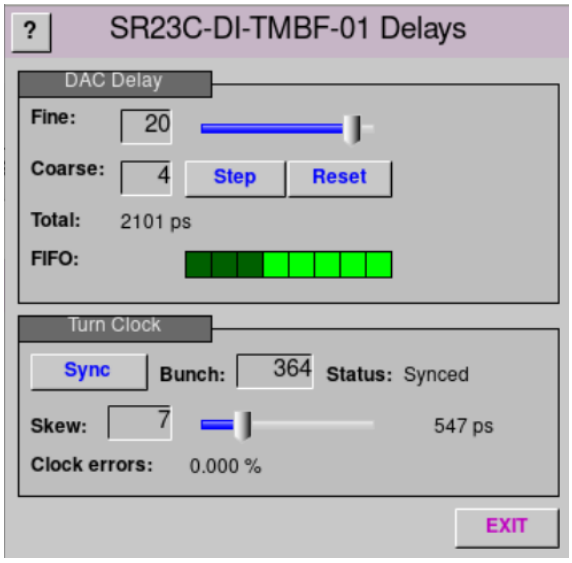


Figure 2.5: DAC delay control screen

Use this setting first to get as close as possible to the target performance. Any remaining adjustment will have to be done with physical cable length changes.

Tuning the cable length

Modify the cable length until the extents of the phase trace comfortably fit within the limits. At

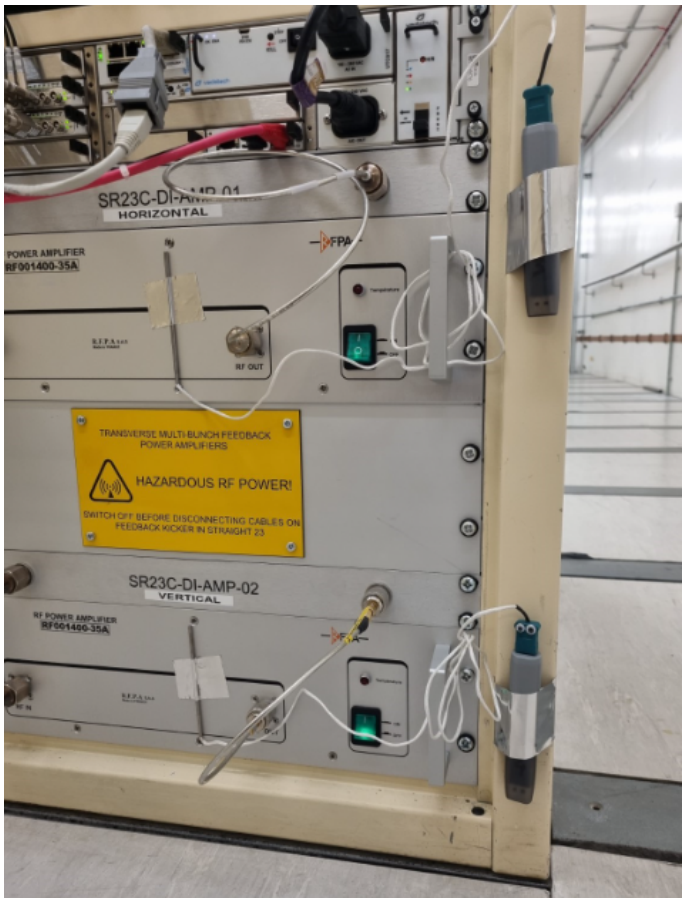


Figure 2.6: Output amplifiers showing the RF cables to adjust

this point the full loop delay should be optimised

2.4 MBF standard operational measurements

2.4.1 Machine setup

- Inject to 30mA 900 bunches (below the instability threshold)

2.4.2 Code to run

The following Matlab code will run modescan, growdamp and spectra measurements for each axis sequentially. The data is stored automatically

```
>> mbf_startup_tests
```

The following explains how to run each separate standard measurement. However there are some common optional inputs.

`'plotting'`, `'yes'` or `'no'`.

Determines if the plotting is displayed. The raw data is saved in either case.

`'auto_setup'`, `'yes'` or `'no'`.

This determines whether to use the main scripts to reset the system to a known state. However this is dangerous if the machine is not in an intrinsically stable condition.

`'tunes'`, previously measured tune value.

The default is to measure the tune as part of the measurement.

Growdamp

The top level code to run a growdamp measurement for each axis is;

```
>> [~] = growdamp_all('x')
>> [~] = growdamp_all('y')
>> [~] = growdamp_all('s')
```

The parameters for the length of excitation and capture are set in `mbf_growdamp_config.m`. The `mbf_growdamp_setup` code sets up the hardware for the correct type of measurement.

`mbf_growdamp_capture` triggers the measurement, captures the data and stores it in the file structure defined in `mbf_system_config`.

Modescan

The top level code to run a modescan measurement for each axis is;

```
>> modescan_all('x')
>> modescan_all('y')
>> modescan_all('s')
```

The `mbf_modescan_setup` code sets up the hardware for the correct type of measurement. `mbf_modescan_capture` triggers the measurement, captures the data and stores it in the file structure defined in `mbf_system_config`.

Spectra

The top level code to run a modescan measurement for each axis is;

```
>> mbf_spectrum_all('x')
>> mbf_spectrum_all('y')
>> mbf_spectrum_all('s')
```

The `mbf_spectrum_setup` code sets up the hardware for the correct type of measurement. `mbf_spectrum_capture` triggers the measurement, captures the data and stores it in the file structure defined in `mbf_system_config`.

2.5 MBF developmental measurements

The MBF system is also used as an instrument for investigating the accelerator behaviour and the behaviour of the beam under different conditions. It can also be used as a driver for certain beam behaviours. These codes tend to be less well tested as they are often only used for particular investigations.

2.5.1 Transverse damping

2.5.2 Emittance control

2.5.3 Bunch motion

`mbf_bunch_motion_setup` sets up all three systems to use the appropriate triggers.

`mbf_bunch_motion_capture` arms the systems, and once triggered by an external trigger, will capture and store the data.

Chapter 3

MBF Archival retrieval

3.1 Introduction

All the MBF codes store their data in a common location with programmatically constructed names. Therefore, it is possible to extract and compare this data.

3.2 Getting the latest data

The simplest thing is to look at the last results taken. To do this simply run the command below on the Matlab command line.

```
>> visualise_latest_mbf_results
```

This will return analysed results for all the standard measurements. For returning results from frontend tuning please see . It is possible to change the x axis of the Growdamp measurement by using the `growdamp_plot_mode` optional input.

```
>> visualise_latest_mbf_results('growdamp_plot_mode', 'freq')
```

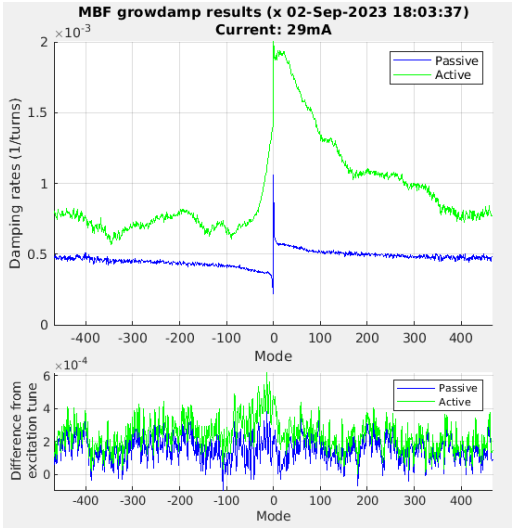
Will put the output on a frequency scale rather than the usual mode scale

```
>> visualise_latest_mbf_results('growdamp_plot_mode', 'pos')
```

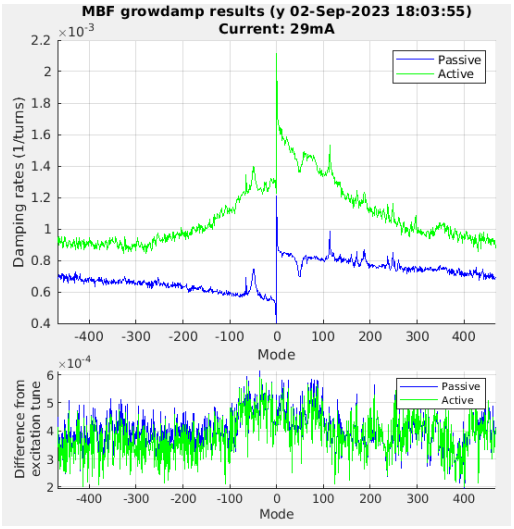
Will reorganise the modes to be numbered 0:935 rather than –468:468

Figure 3.1 is an example of the output for Growdamp, figure 3.2 of Modescan measurements and figure 3.3 of spectral measurements.

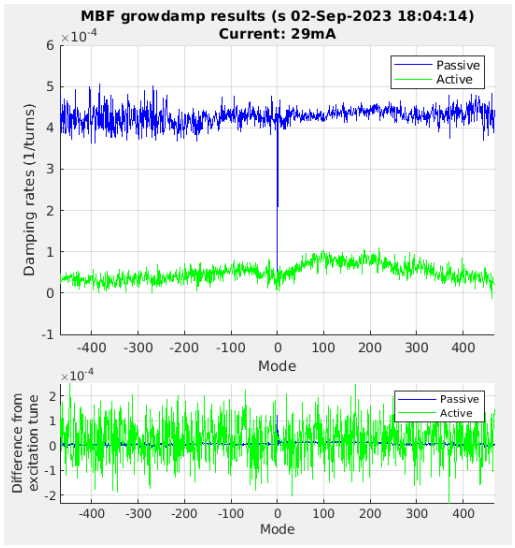
The spectra graphs show the residual motion after all feedbacks have been applied.



(a) Horizontal axis

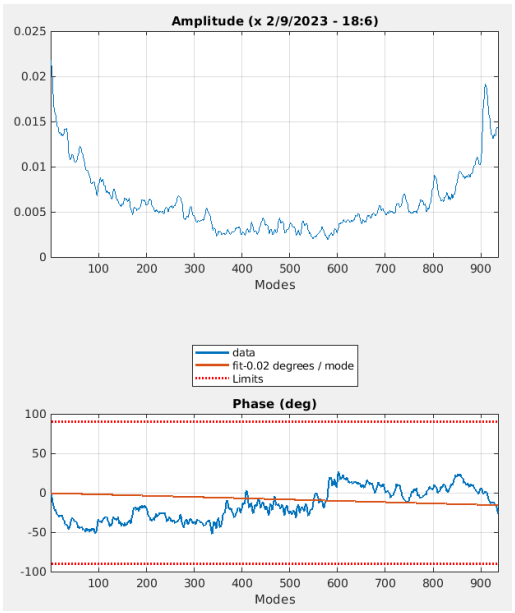


(b) Vertical axis

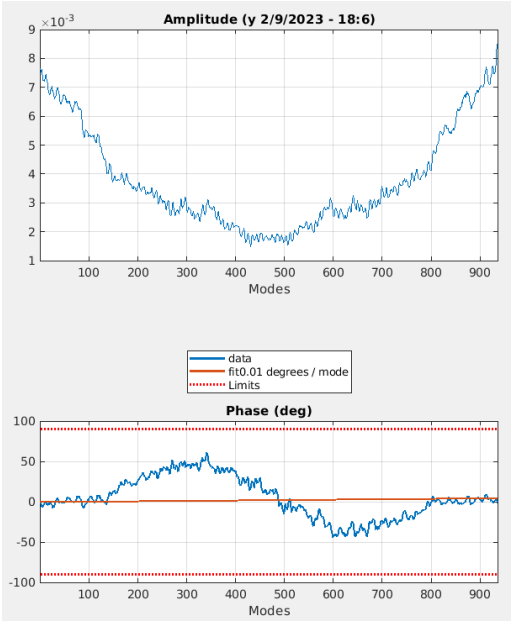


(c) longitudinal axis

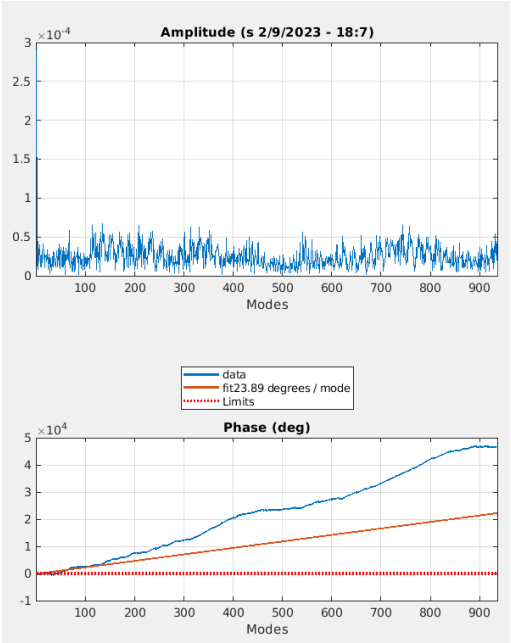
Figure 3.1: Example Growdamp results



(a) Horizontal axis

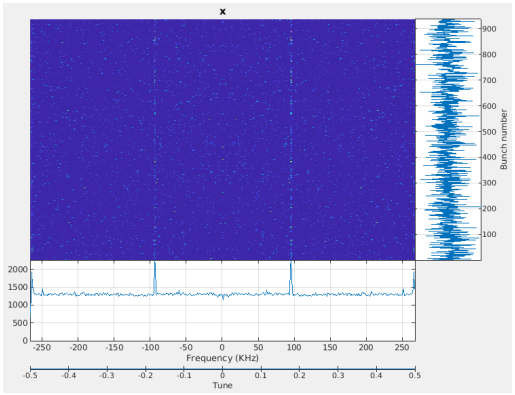


(b) Vertical axis

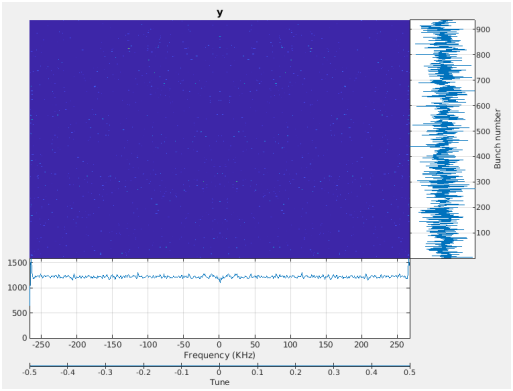


(c) longitudinal axis

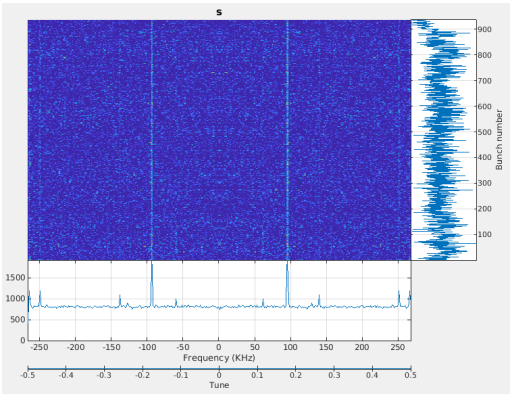
Figure 3.2: Example modescan results



(a) Horizontal axis



(b) Vertical axis



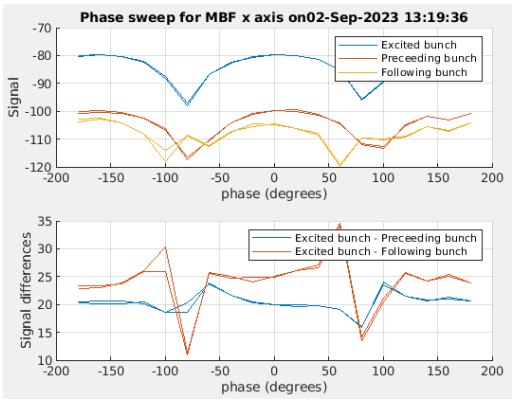
(c) longitudinal axis

Figure 3.3: Example spectrum results- PLACEHOLDER

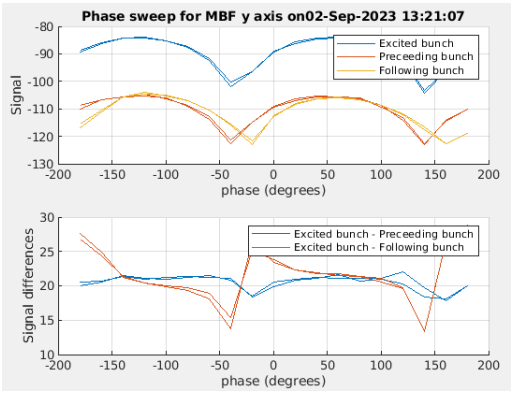
3.3 Frontend tuning results

By passing in some optional setting to `visualise_latest_mbf_results` it will output the results from the latest results from the DORIS and frontend tuning.

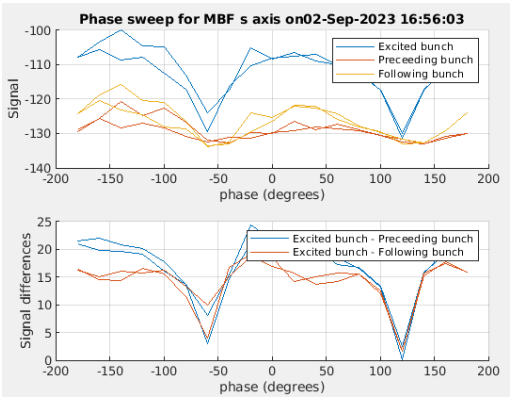
```
>> visualise_latest_mbf_results('result_type', 'frontend')
```



(a) Horizontal axis

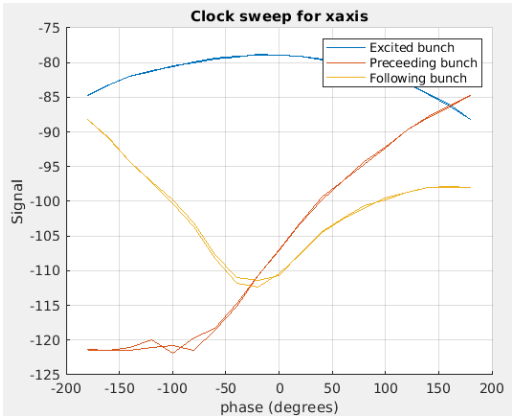


(b) Vertical axis



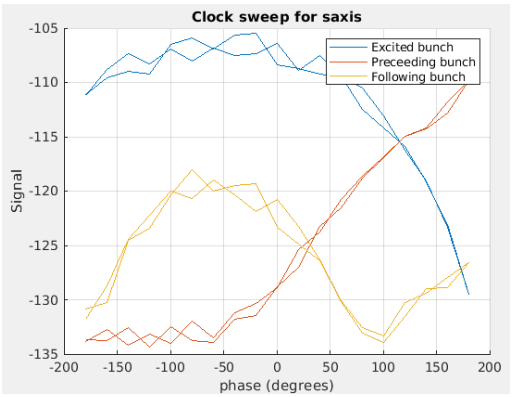
(c) longitudinal axis

Figure 3.4: Example system phase results



(a) Horizontal axis

(b) Vertical axis



(c) longitudinal axis

Figure 3.5: Example clock phase results

3.4 Getting any historic data

A more flexible way is to set up the time period you are interested in looking at and use the measurement specific retrieval tool.

You can set up the period of interest by using a modified version the following example code.

```
>> start_datenum = datetime('16/04/2023_21:38',...
    'InputFormat','dd/MM/yyyy_HH:mm');
>> end_datenum = datetime('16/06/2023_22:55',...
    'InputFormat','dd/MM/yyyy_HH:mm');
>> date_range = [start_datenum, end_datenum];
```

Starting with growdamp data, the following command will extract any measurements taken within the date range for the y axis and then plot them on top of one another. An example of this is shown in figure 3.6.

```
>> conditioned_data = mbf_growdamp_archival_retrieval('y',...
    date_range);
```

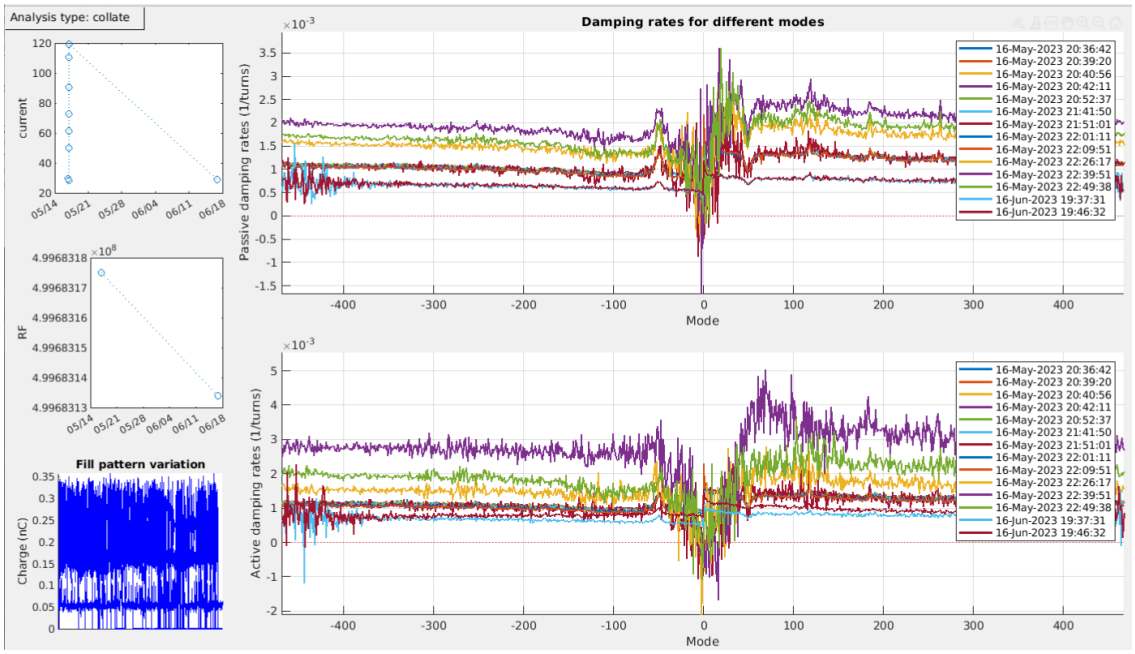


Figure 3.6: Collated growdamp results

This allows easy identification of changes in behaviour

As machine conditions can be different it is possible to filter on current range and RF frequency range

For example, by setting the `current_range` optional input we can select only measurements which were taken between 2mA and 40mA (figure 3.7).

```
>> conditioned_data = mbf_growdamp_archival_retrieval('y',...
    date_range, 'current_range', [2 40]);
```

This is often useful to ensure that any measurements used are below instability thresholds. One common set of measurements beyond simple comparisons is plotting the measurements against a changing machine parameter (usually current)

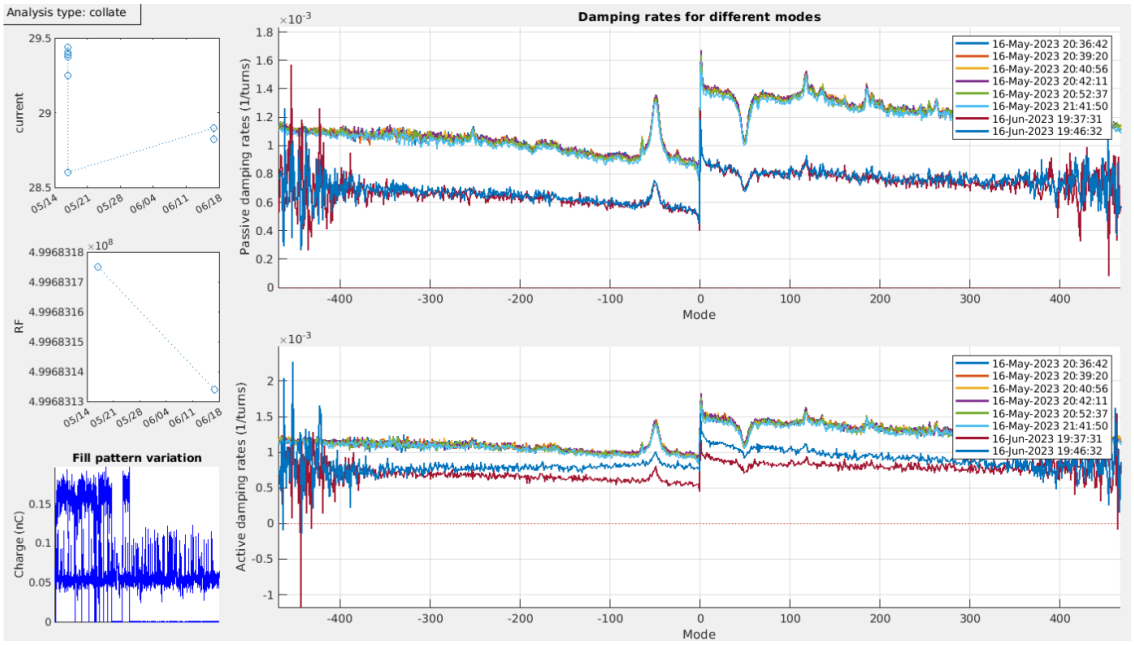


Figure 3.7: Collated growdamp results only returning results which are within the specified current range.

Although it is possible to extract such relationships from the data just using the variation over time, it is more usual to have a dedicated measurement set for the purpose. In this case you would set the period to only contain data from that experiment. Then you would run the retrieval with the analysis type set to sweep and then set the appropriate parameter to sweep over (again it is usually current but any variable in the datasets can be used). Figure 3.8 shows an example of this.

```
>> conditioned_data = mbf_growdamp_archival_retrieval('y', ...
    date_range, 'analysis_type', 'sweep',...
    'sweep_parameter', 'current');
```

At this point further analysis is possible. In the case of a current sweep, it is possible to predict the current which individual modes will go unstable. (figure 3.9).

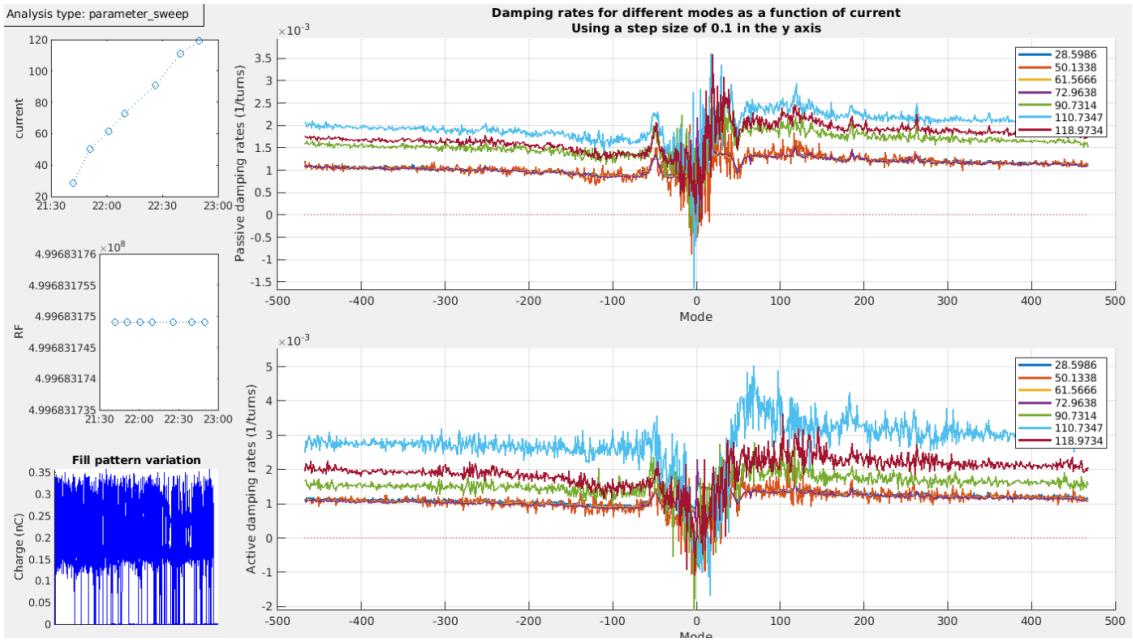


Figure 3.8: Growdamp results against a swept parameter. In this case the beam current. Note: These are placeholder graphs.

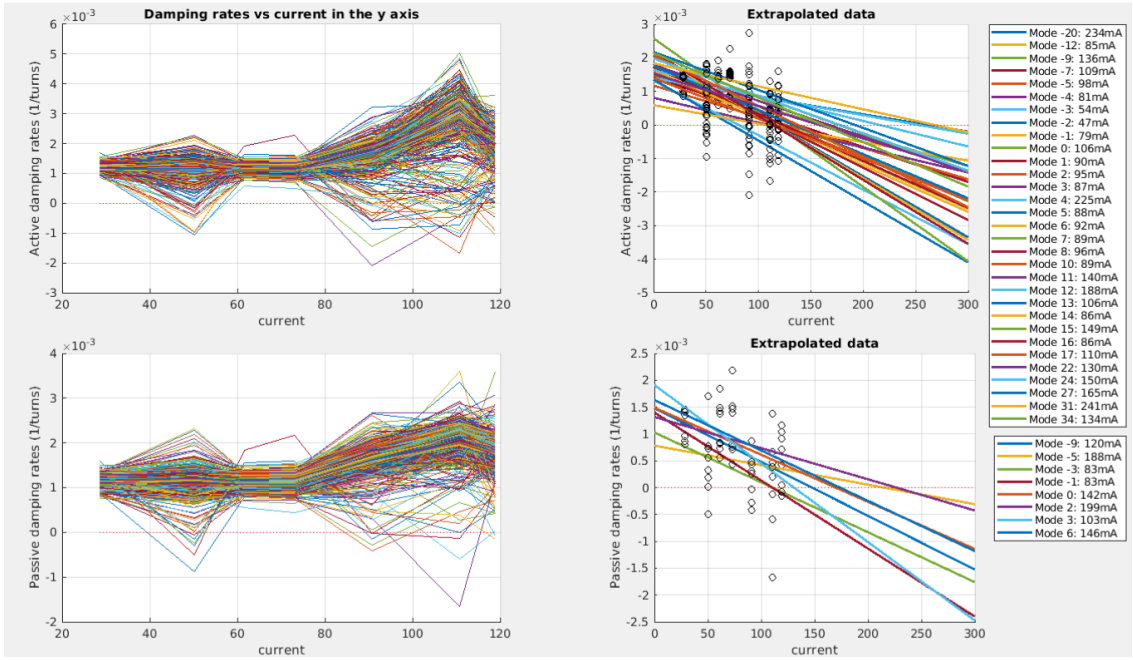


Figure 3.9: Results from further analysis of the current dependent growdamp data. Note: These are placeholder graphs.

Modescan data retrieval is much the same as the growdamp retrieval. Once the date range is set up the following code will extract and display all results from the y axis (figure 3.10).

```
>> conditioned_data = mbf_modescan_archival_retrieval('y',...
    date_range);
```

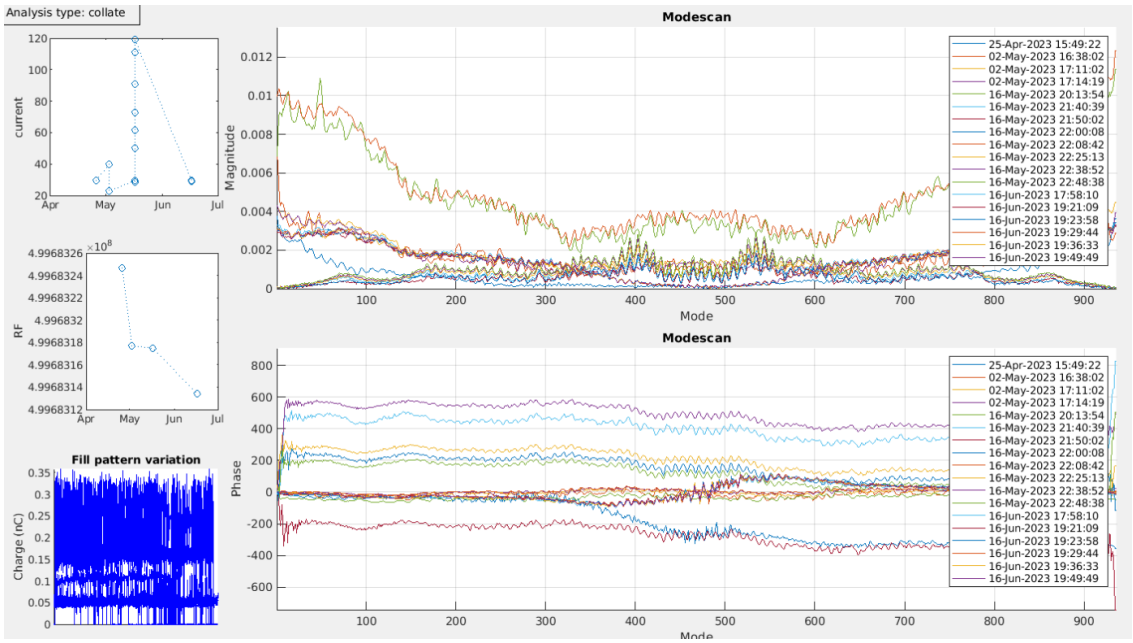


Figure 3.10: Collated modescan results

As before you can filter on current range(figure 3.11).

```
>> conditioned_data = mbf_modescan_archival_retrieval('y',...
    date_range, 'current_range', [2 40]);
```

And can plot against a parameter sweep (figure 3.12).

```
>> conditioned_data = mbf_modescan_archival_retrieval('y',...
    date_range, 'analysis_type', 'sweep',...
    'sweep_parameter', 'current');
```

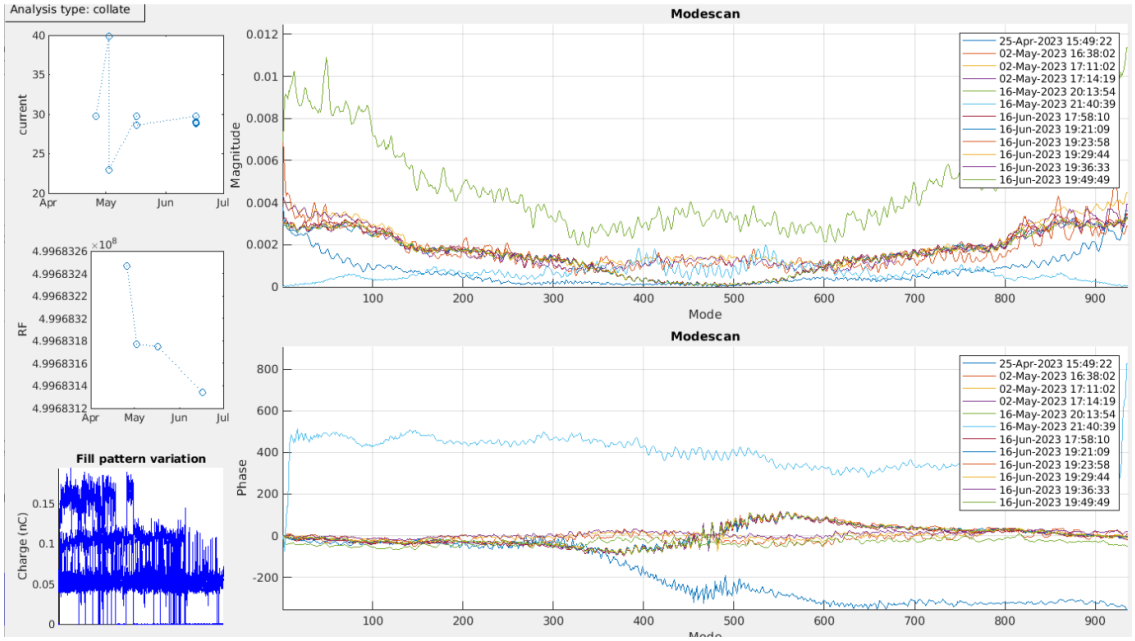


Figure 3.11: Collated modescan results restricted to the specified current range.

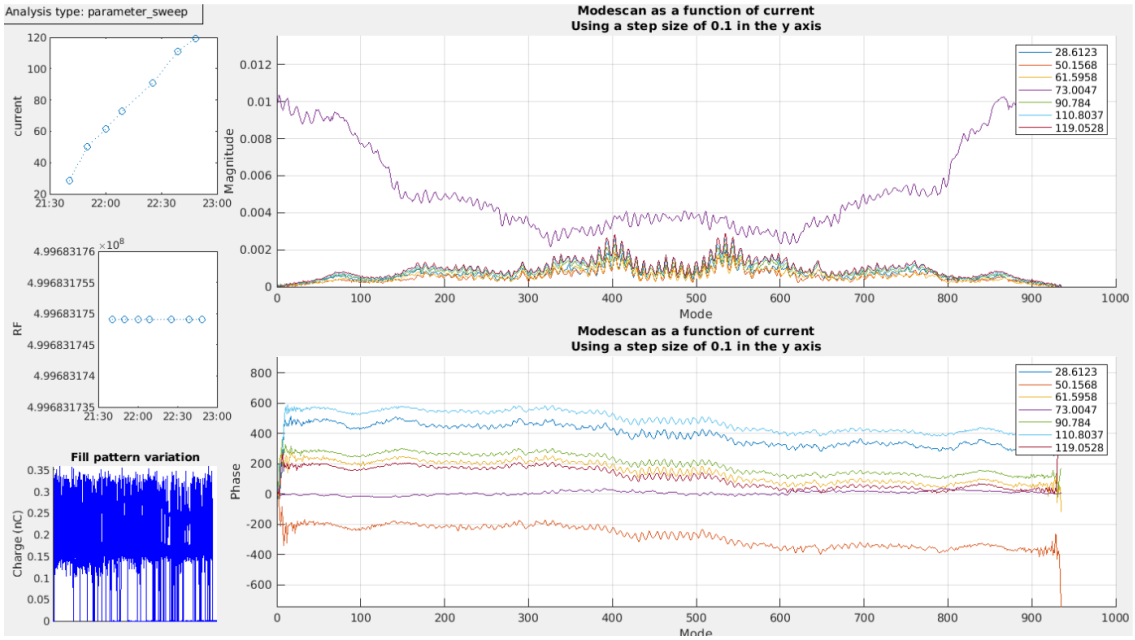


Figure 3.12: Growdamp results against a swept parameter. In this case the beam current.

Spectra
Tune sweep over modes