End-of-Year
LHC Beam-Based Feedbacks
Software Overview

Ralph J. Steinhagen, BE-BI

Some references:

http://cern.ch/AB-seminar/talks/AB.Seminar.rst.pdf (CERN-AB-2007-049)

http://lhccwg.web.cern.ch/lhccwg/Meetings/2007/2007.10.23/2007-10-23_LHCCWG-FAULTY_BPM.pdf

http://lhc-beam-operation-committee.web.cern.ch/lhc-beam-operation-committee/minutes/Meeting25_29_11_2011/2011-11-29_LBOC_OrbitFB_Bandwidth.pdf

http://accelconf.web.cern.ch/AccelConf/PAC2011/talks/weobn2_talk.pdf &

http://accelconf.web.cern.ch/AccelConf/PAC2011/papers/weobn2.pdf

LHC-BPM-ES-0004 rev. 2.0, EDMS #327557, 2002,

svn+ssh://svn.cern.ch/reps/acco-co/trunk/lhc/lhc-feedbacks – or –

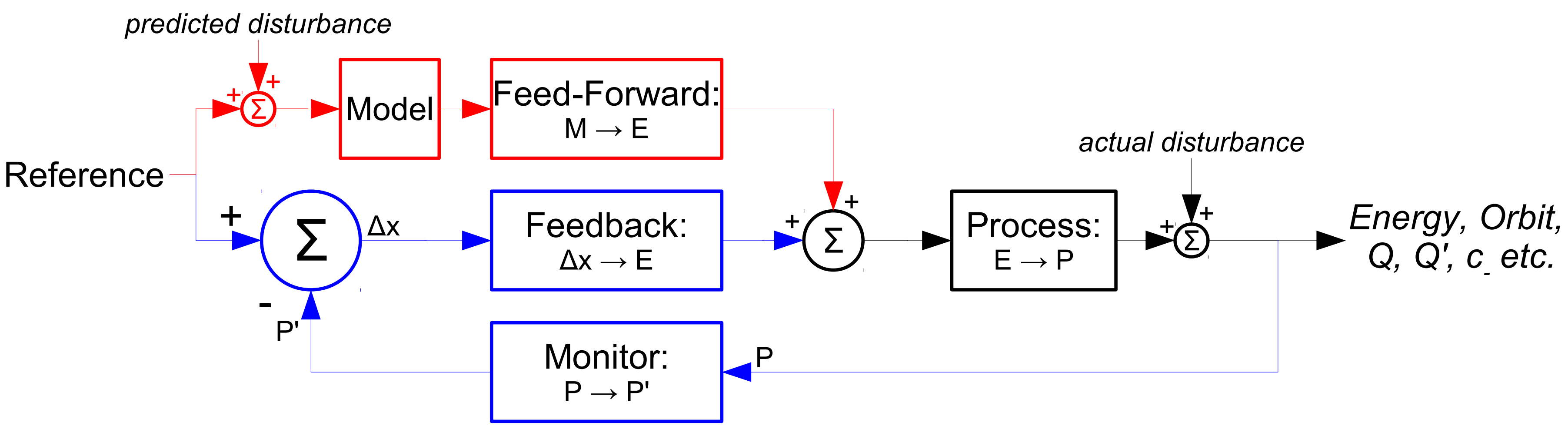
http://sources/browse/acc-co/trunk/lhc/lhc-feedbacks

Overview

- The BIG WHY?
- Feedback Function and Architecture
- Why the OFC is using CERN's ROOT framework
- Architecture and where to find the source code documentation
- Status and Outlook for Expert Java application
- Brief: what needs to be tackled for 2012

Parameter control, either through...

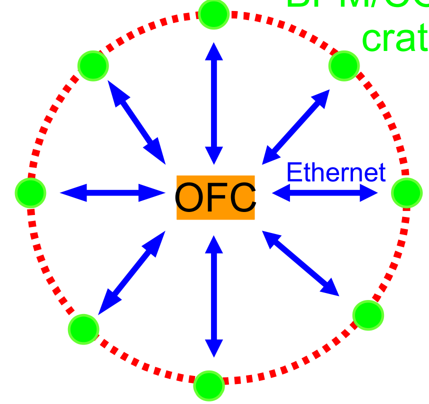
- Feed-Forward: (FF)
  - Steer parameter using precise process model and disturbance prediction
- Feedback: (FB)
  - Steering using rough process model and measurement of parameter
  - Two types: within-cycle (repetition Δt<<10 hours) or cycle-to-cycle (Δt>10 hours)



  - Both do not mix well if the FB is not the slave of the FF, paradigm change:
    - Feed-Forward: trims the actual parameter (e.g. PC currents)
    - Feedback: trim the parameter reference

LHC: orbit feedback system

- Small perturbations around the reference orbit will be continuously compensated using beam-based alignment through a central global orbit feedback with local constraints:



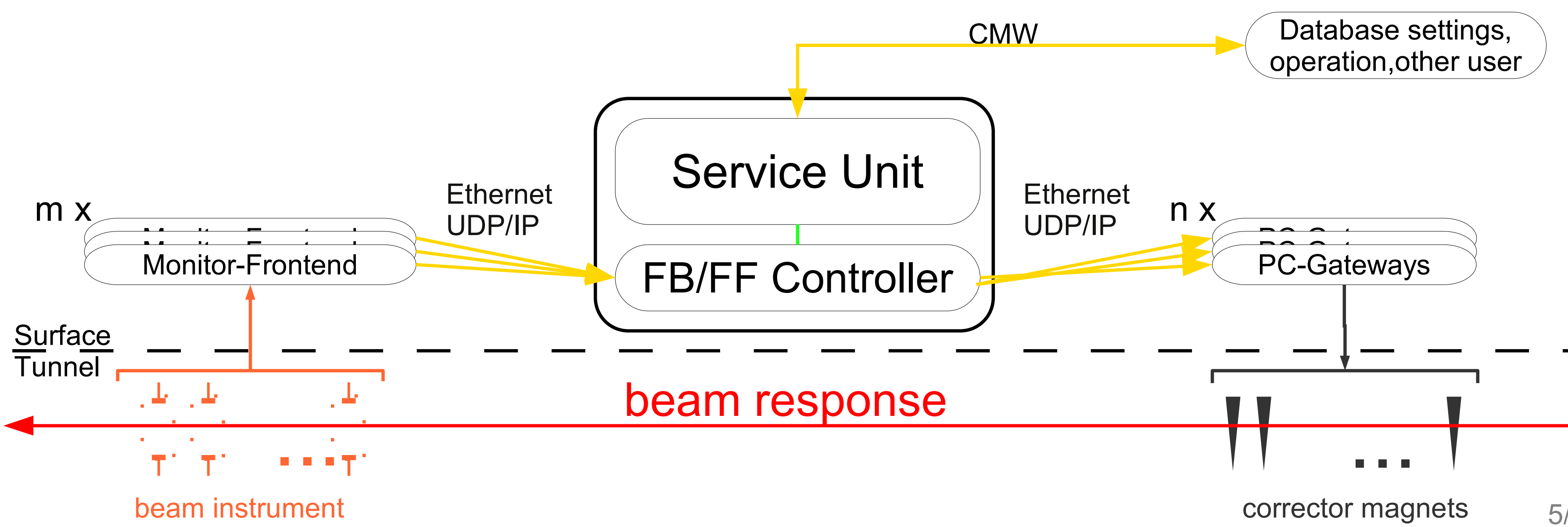
  - 1070 beam position monitors
    - BPM spacing: $\Delta\mu_{BPM} \approx 45°$ (oversampling → robustness!)
    - Measure in both planes: > 2140 readings!
  - One Central Orbit Feedback Controller (OFC)
    - Gathers all BPM measurements, computes and sends currents through Ethernet to the PC-Gateways to move beam to its reference position:
      - high numerical and network load on controller front-end computer
      - a rough machine model is sufficient for steering (insensitive to noise and errors)
      - most flexible (especially when correction scheme has to be changed quickly)
      - easier to commission and debug
  - 530 correction dipole magnets/plane (71% are of type MCBH/V, ±60A)
    - total 1060 individually powered magnets (60-120 A)
    - ~30 shared between B1&B2

- With more than 3100 involved devices the largest and most complex system

Common Feedback/Feed-forward Control Layout

LHC feedback control scheme implementation split into two sub-systems:

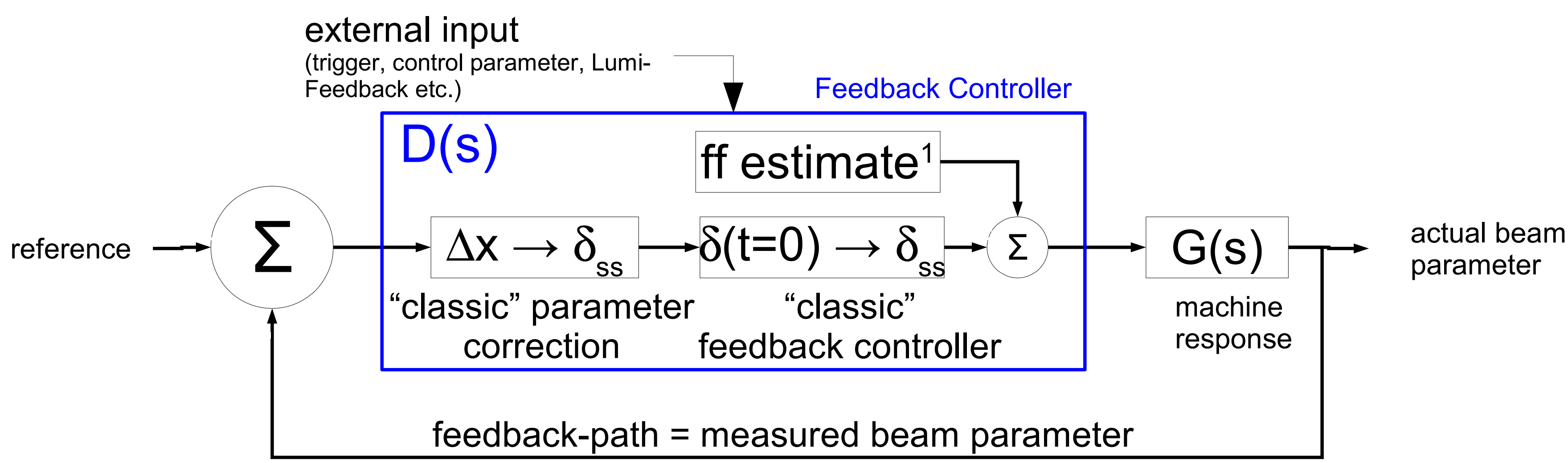
- Feedback Controller: actual parameter/feedback controller logic
  - Simple streaming task for all feed-forwards/feedbacks:
    (Monitor → Network $)_{FB}$→ Data-processing → Network → PC-Gateways
  - real-time operating system, constant load, can run auto-triggered
  - Initially targeted to be on an FPGA for reliability reasons
- Service Unit: Interface to users/software control system



Multiple-Input-Multiple-Output (MIMO) Process Control

- 'Divide and Conquer' feedback controller design approach:
  1. Compute steady-state corrector settings $\vec{\delta}_{ss} = (\delta_1, \ldots, \delta_n)$ based on measured parameter shift $\Delta x = (x_1, \ldots, x_n)$ that will move the beam to its reference position for $t \to \infty$. — space domain
  2. Compute a $\vec{\delta}(t)$ that will enhance the transition $\vec{\delta}(t=0) \to \vec{\delta}_{ss}$
  3. Feed-forward: anticipate and add deflections $\vec{\delta}_{ff}$ to compensate changes of well known and properly described sources

  (2 and 3: time domain)



- (N.B. here G(s) contains the process and monitor response function)

- Effects on orbit, Energy, Tune, Q' and C⁻ can essentially cast into matrices:

$$\Delta\vec{x}(t)=\underline{R}\cdot\vec{\delta}(t) \quad with \quad R_{ij}=\frac{\sqrt{\beta_i\beta_j}}{2\sin(\pi Q)}\cdot\cos(\Delta\mu_{ij}-\pi Q)+\frac{D_i D_j}{C(\alpha_c-1/\gamma^2)}$$

matrix multiplication

  - LHC matrices' dimensions:

$$\underline{R}_{orbit}\in\mathbb{R}^{1070\times 530} \quad \underline{R}_{Q}\in\mathbb{R}^{2\times 16} \quad \underline{R}_{Q'}\in\mathbb{R}^{2\times 32} \quad \underline{R}_{C^-}\in\mathbb{R}^{2\times 10/12}$$

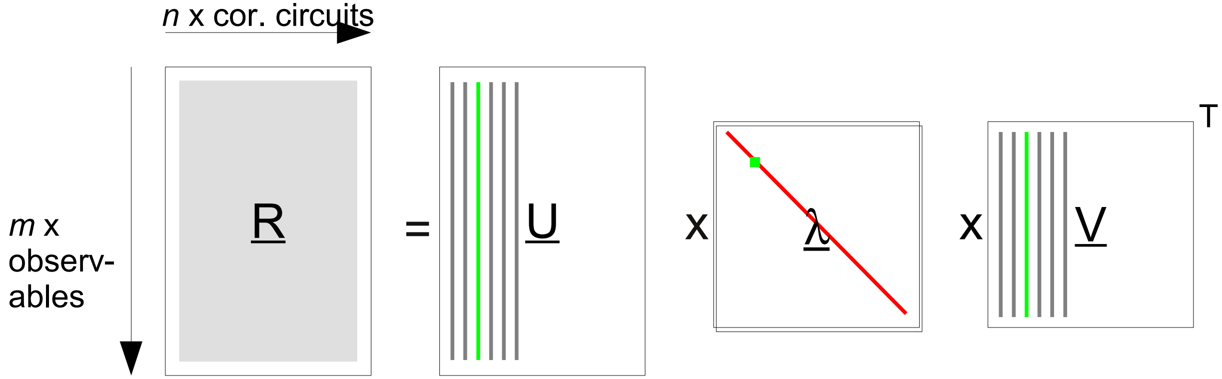
  - control consists essentially in inverting these matrices:

$$\left\|\vec{x}_{ref}-\vec{x}_{actual}\right\|_2=\left\|\underline{R}\cdot\vec{\delta}_{ss}\right\|_2<\epsilon \;\rightarrow\; \vec{\delta}_{ss}=\tilde{R}^{-1}\Delta\vec{x}$$

- Some potential complications:
  - Singularities = over/under-constraint matrices, noise, element failures, spurious BPM offsets, calibrations, ...
  - Time dependence of total control loop → “The world goes SVD....”

Space-Domain: Singular Value Decomposition (SVD) on a slide

Linear algebra theorem*:



eigen-vector relation:

$$\lambda_i \vec{u}_i = \underline{R} \cdot \vec{v}_i$$

$$\lambda_i \vec{v}_i = \underline{R}^T \cdot \vec{u}_i$$

- though decomposition is numerically more complex final correction is a simple vector-matrix multiplication:

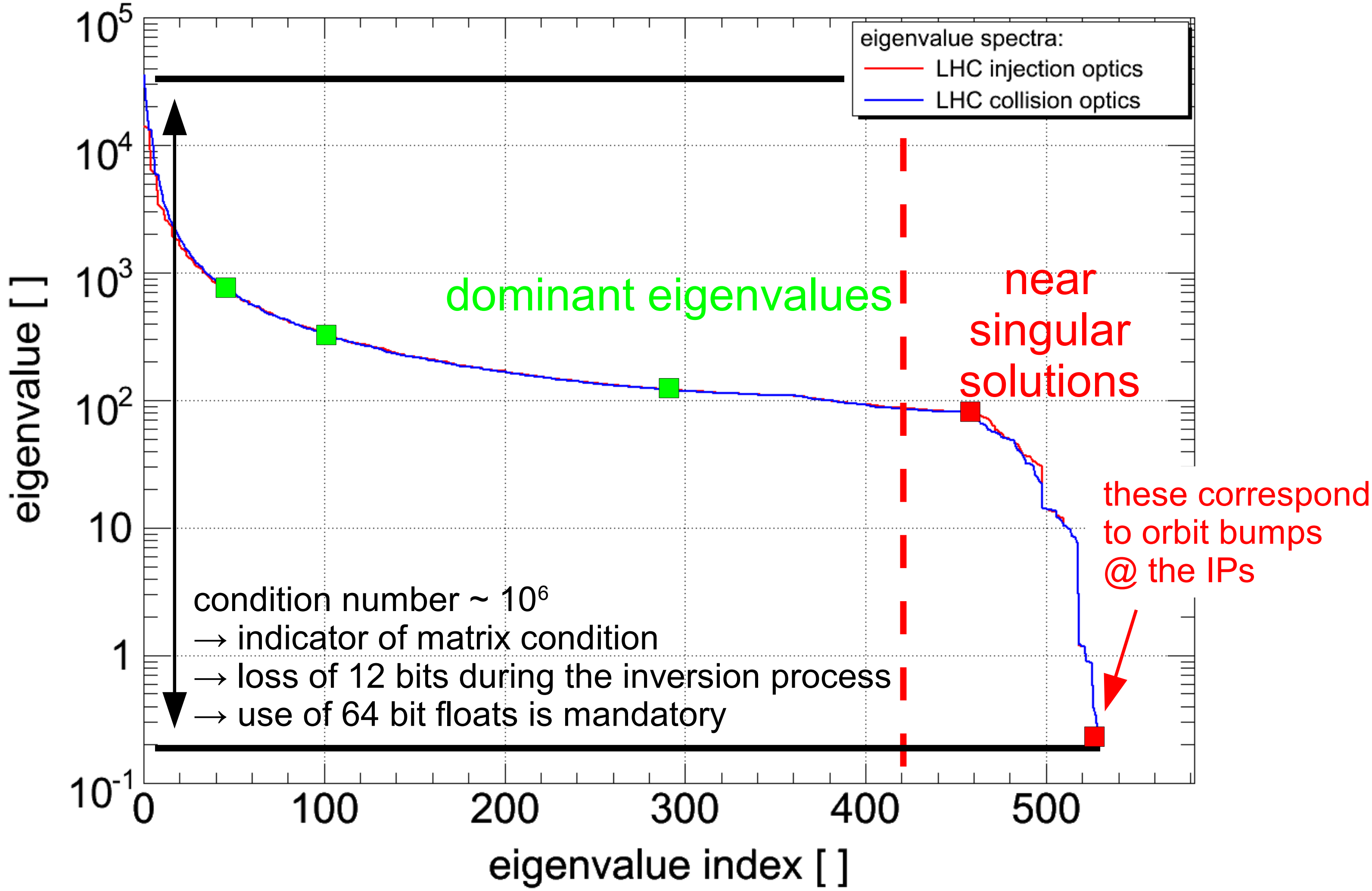
$$\vec{\delta}_{ss} = \tilde{R}^{-1} \cdot \Delta \vec{x} \quad with \quad \tilde{R}^{-1} = \underline{V} \cdot \underline{\lambda}^{-1} \cdot \underline{U}^T \quad \Leftrightarrow \quad \vec{\delta}_{ss} = \sum_{i=0}^{n} \frac{a_i}{\lambda_i} \vec{v}_i \quad with \quad a_i = \vec{u}_i^T \Delta \vec{x}$$

- numerical robust, minimises parameter deviations Δx and circuit strengths δ
- Easy removal of singularities, (nearly) singular eigen-solutions have $\lambda_i \sim 0$
  - to remove those solution: if $\lambda_i \approx 0 \rightarrow$ '$1/\lambda_i := 0$'
  - discarded eigenvalues corresponds to solution pattern unaffected by the FB

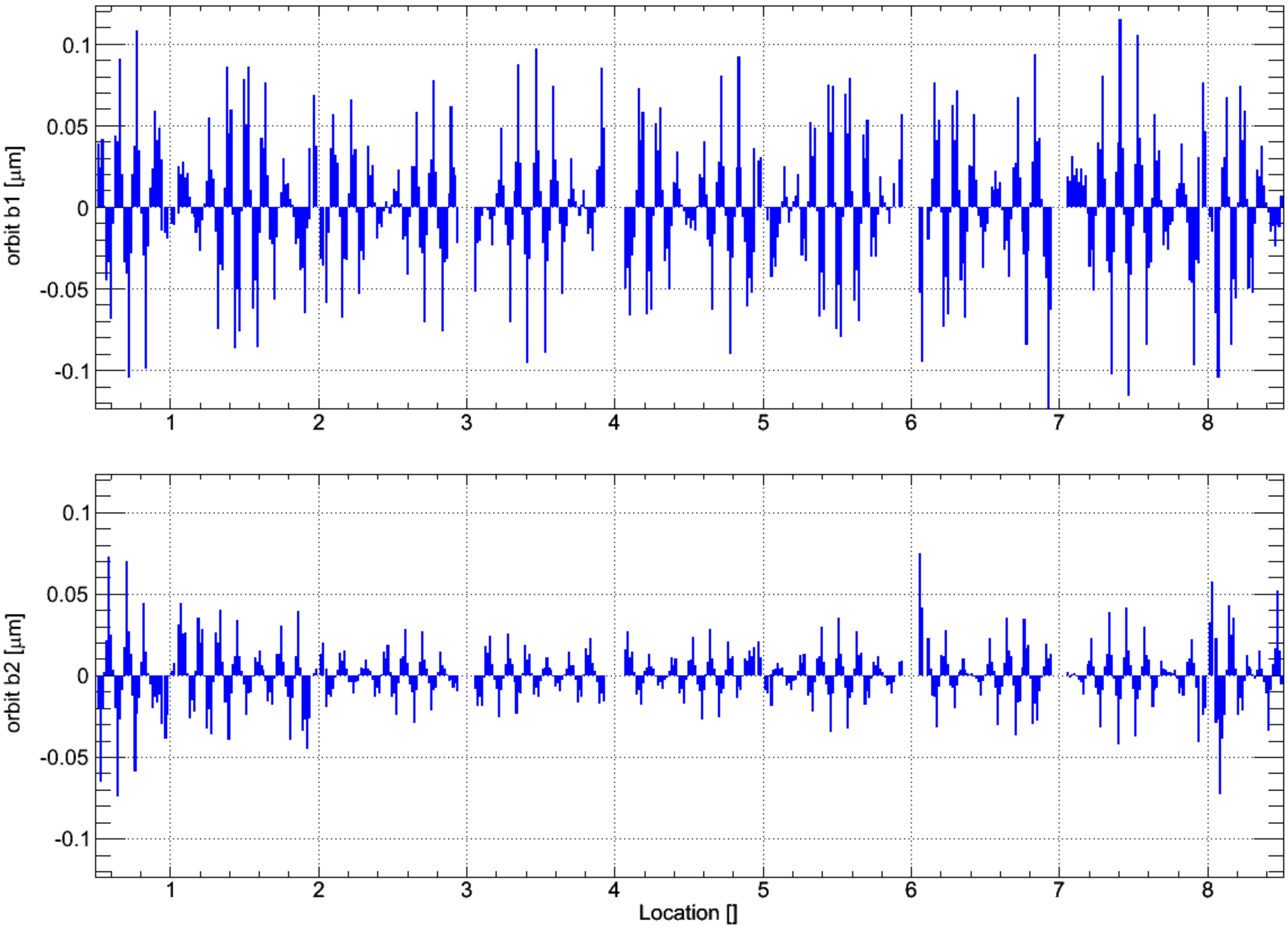
*G. Golub and C. Reinsch, "*Handbook for automatic computation II, Linear Algebra*", Springer, NY, 1971

Space-Domain:
SVD example: LHC eigenvalue spectrum

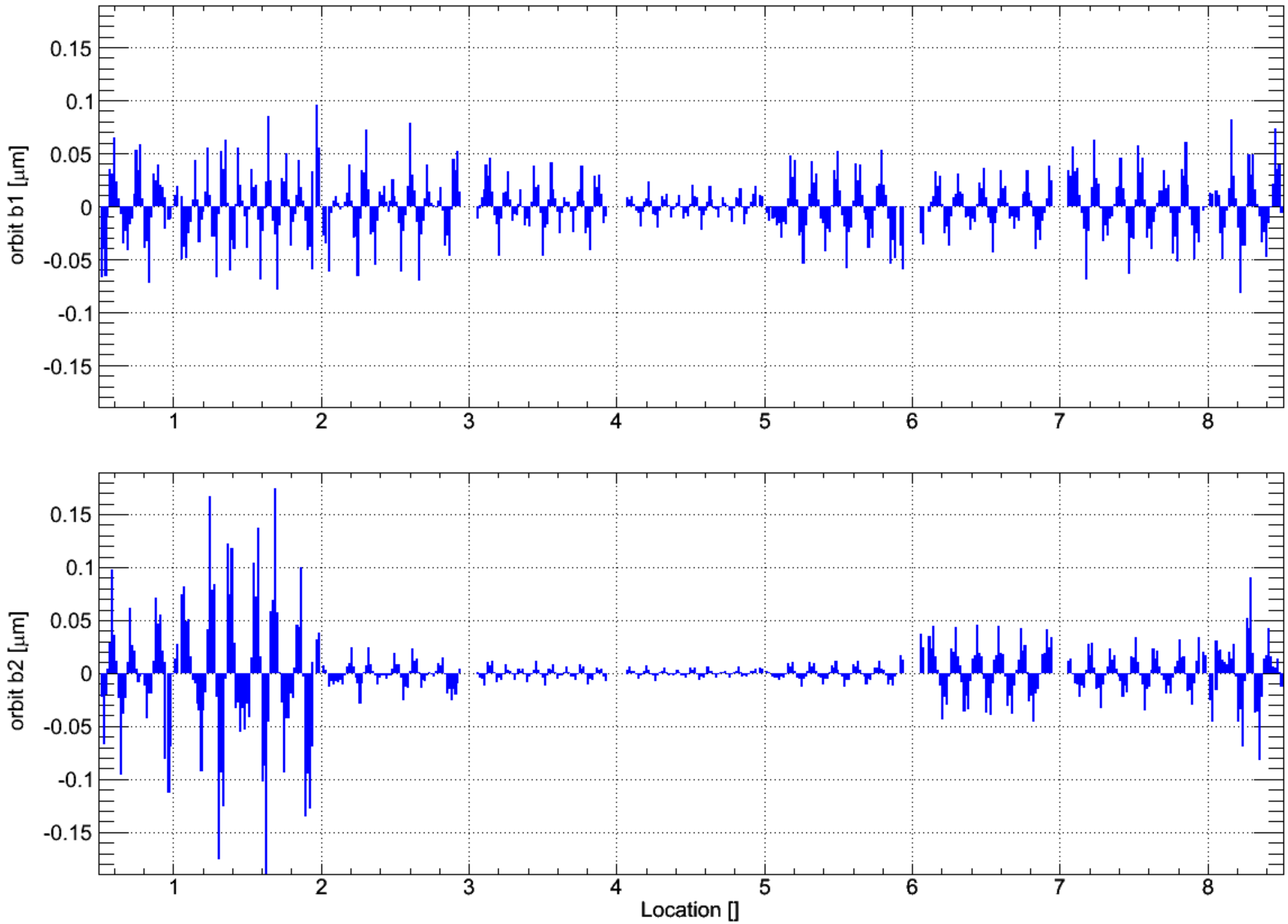
Eigenvalue spectra for vertical LHC response matrix using all BPMs and CODs:



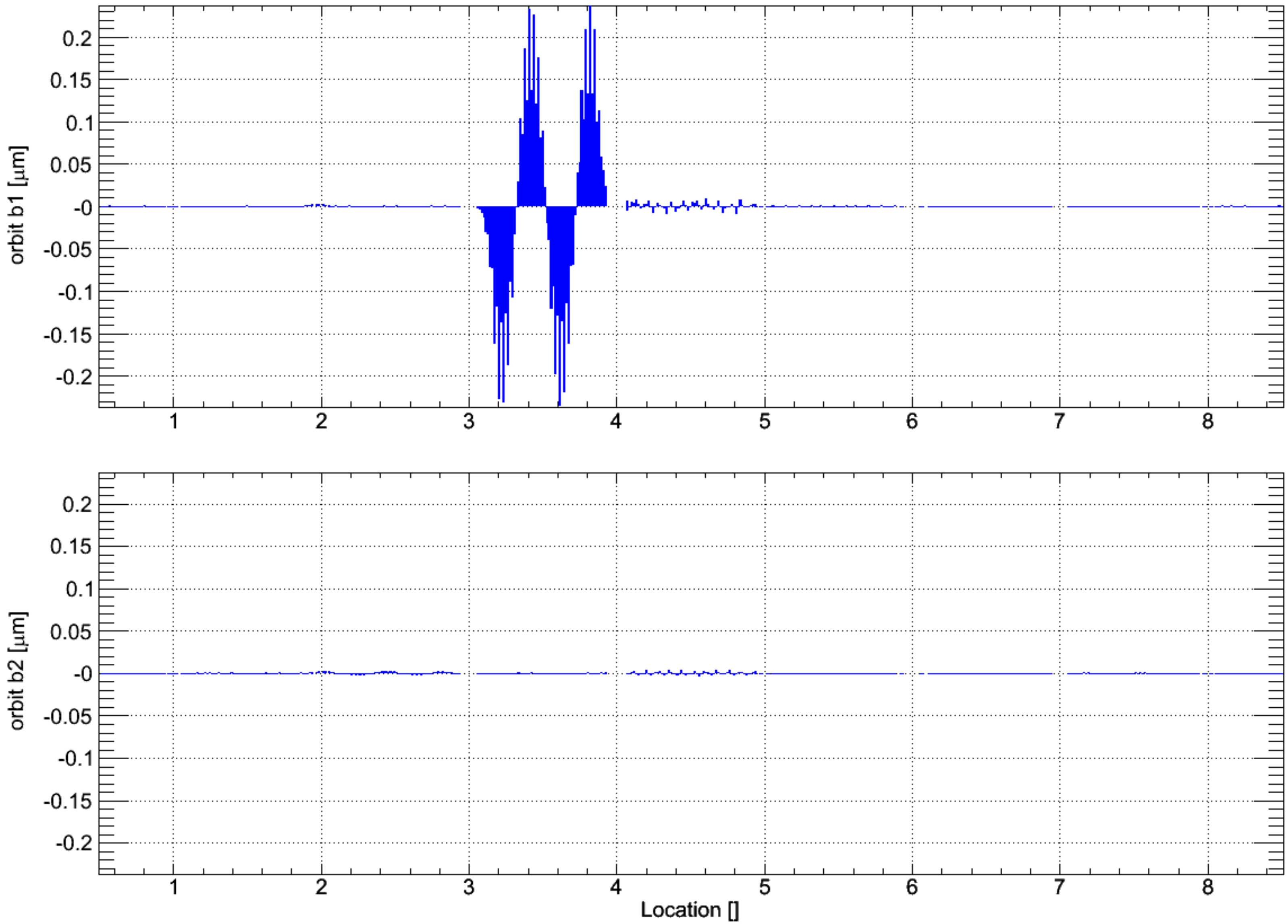
Space Domain: LHC BPM eigenvector #50 $\lambda_{50} = 6.69 \cdot 10^2$



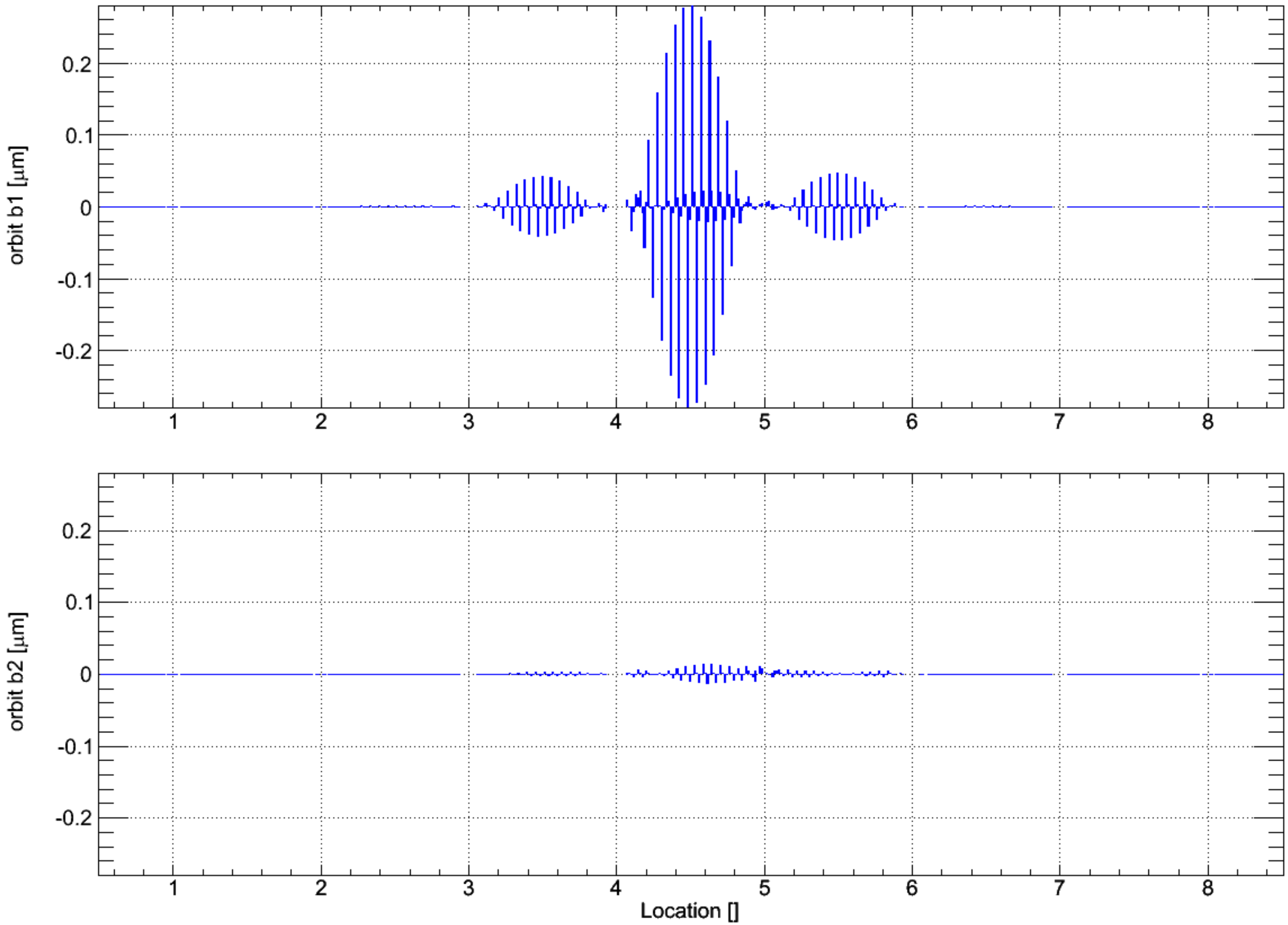
Space Domain:
LHC BPM eigenvector #100 λ_{100}= 3.38•10^2



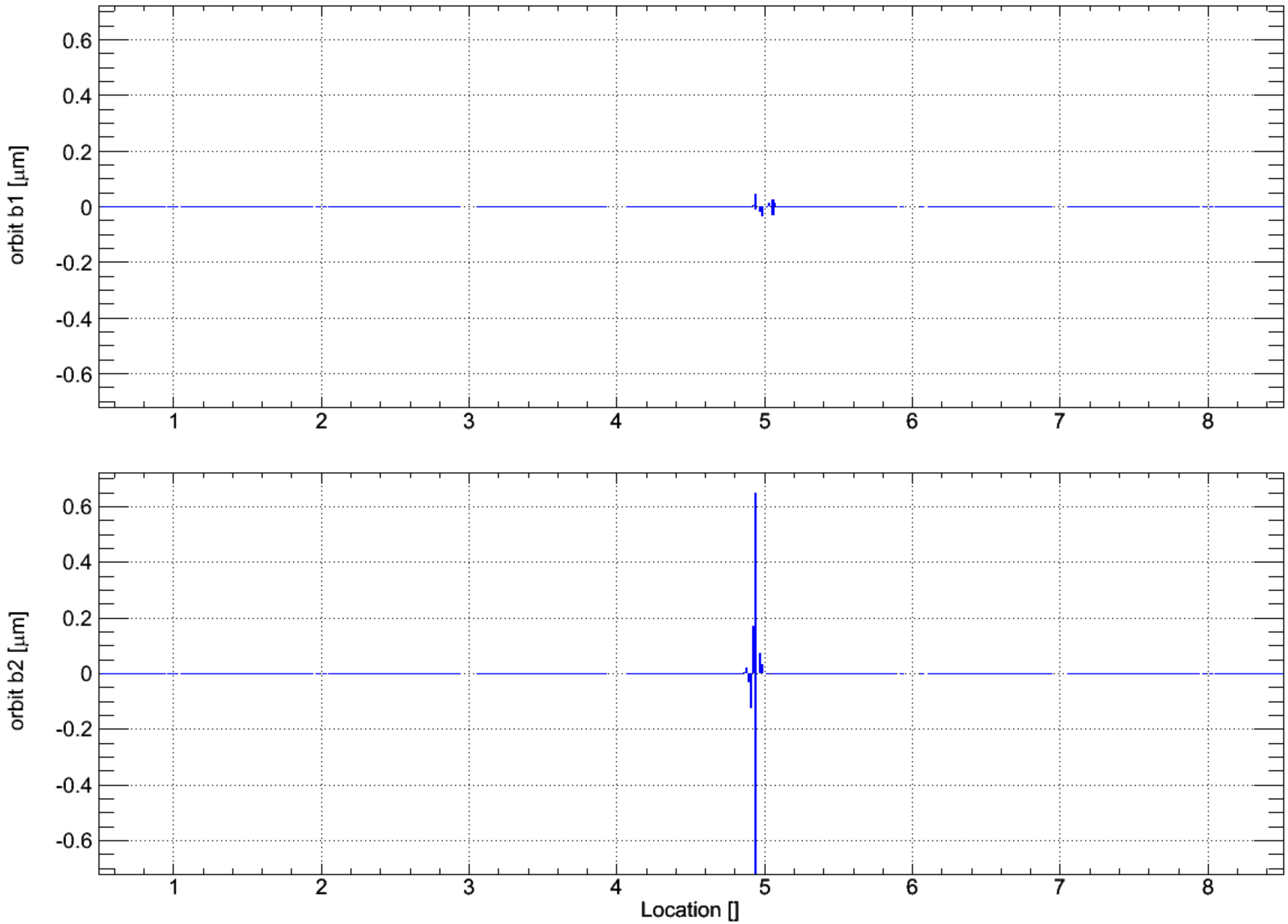
Space Domain:
LHC BPM eigenvector #291 λ_{291}= 2.13•10^2



Space Domain:
LHC BPM eigenvector #449 λ_{449}= 8.17•10^1



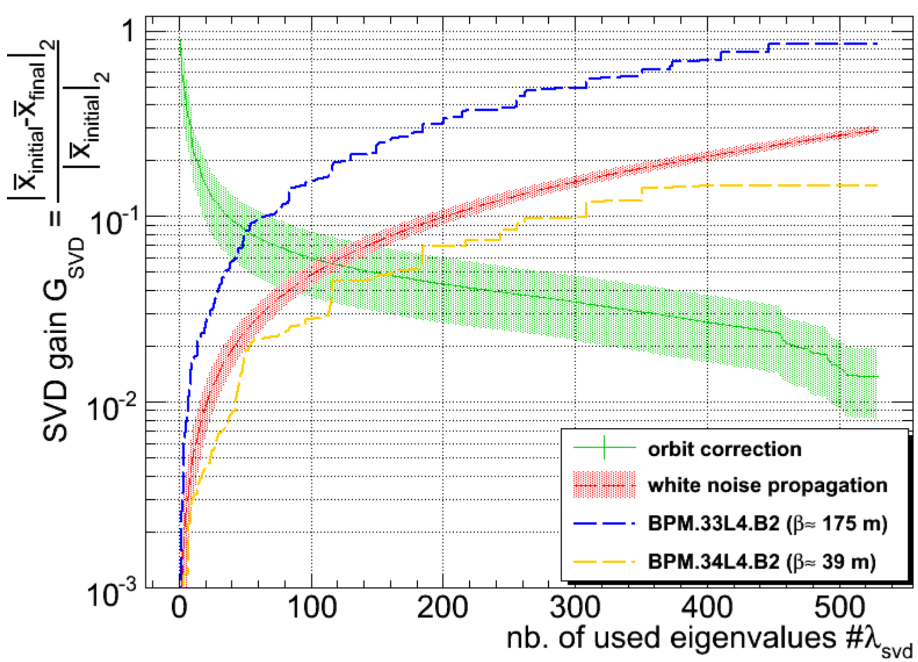
Space Domain: LHC BPM eigenvector #521 $\lambda_{521} = 1.18 \cdot 10^{0}$

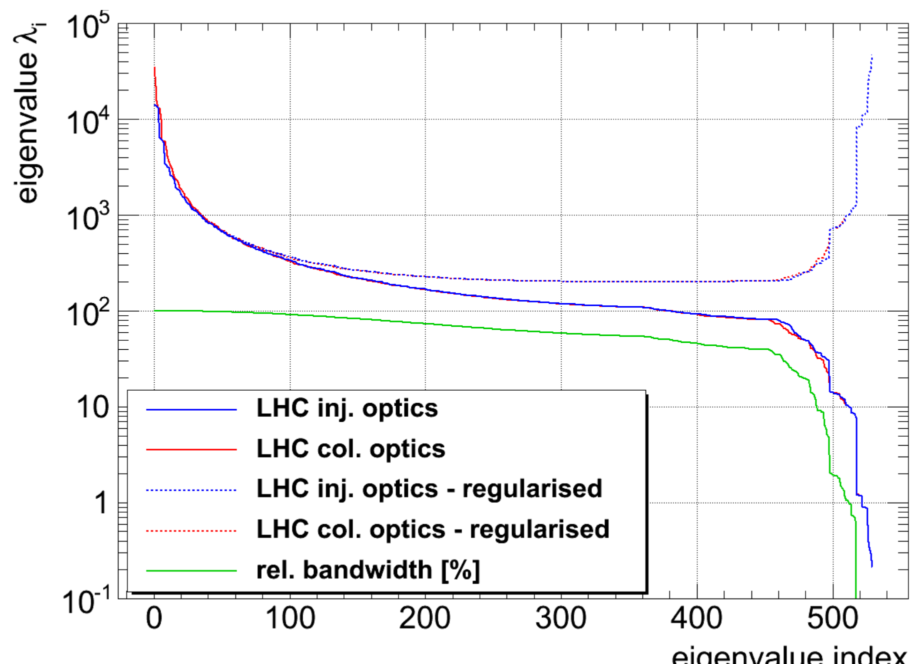


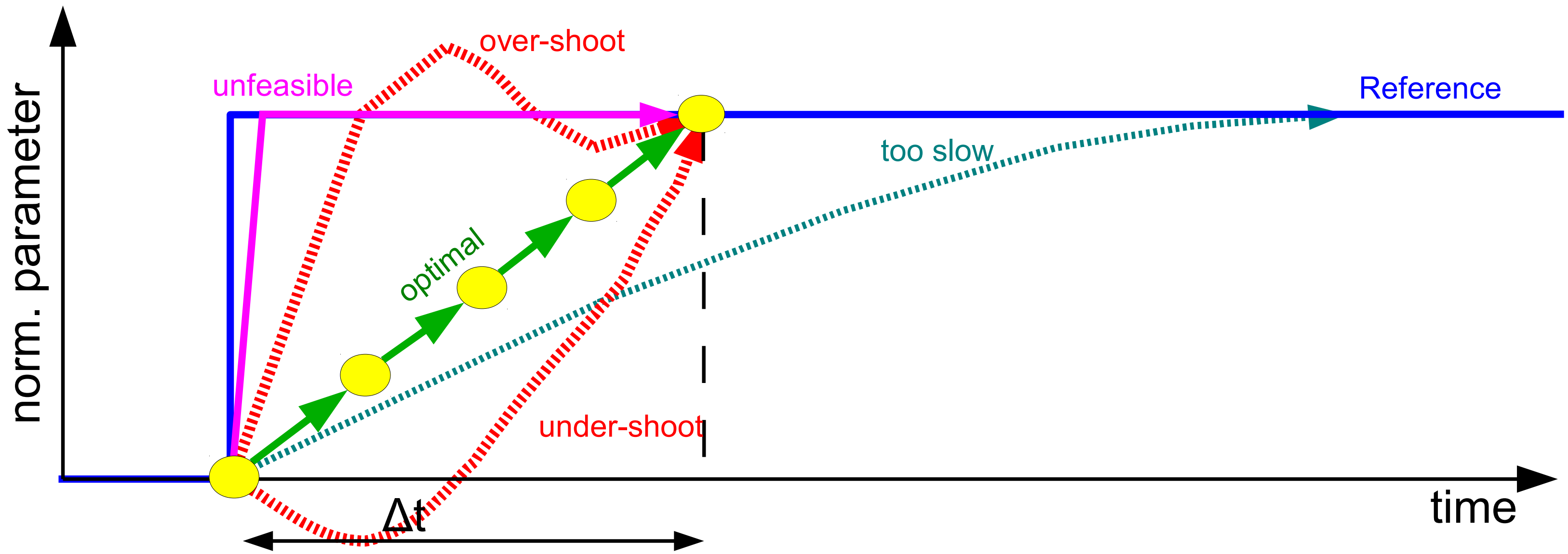
Standard Singular-Value-Decomposition based Orbit Correction

- Initially: Truncated-SVD (set $\lambda_i^{-1} := 0$, for i>N)
  - not without issues: removed λ_i allowed local bumps creeping in (e.g. collimation)

- Regularised-SVD (Tikhonov/opt. Wiener filter with $\lambda_i^{-1} := \lambda_i/(\lambda_i^2 + \mu)$, $\mu > 0$)
  - more robust w.r.t. optics errors and mitigation of BPM noise/errors
    → allowed re-using same ORM for injection, ramp and 10+ squeeze steps







- Optimal control [or design] ...

  "... deals with the problem of finding a control law for a given system such that a given optimality criterion is achieved. A control problem includes a cost functional that is a function of state and control variables."

  – Common criteria: closed loop stability, minimum bandwidth, minimisation of action integral, power dissipation, ...

- classic closed loop: $T_0(s) = \frac{D(s)G(s)}{1+D(s)G(s)}$ → "this tells me???"

- Using Youla's method: “design closed loop in a open loop style”:
- Youla showed[1] that all stable closed loop controllers D(s) can be written as:

$$D(s)=\frac{Q(s)}{1-Q(s)G(s)} \qquad (1)$$

- Example: first order system

$$G(s)=\frac{K_0}{\tau s+1} \quad \text{with } \tau \text{ being the circuit time constant} \qquad (2)$$

- Using for example the following ansatz:

$$Q(s)=F_Q(s)G^i(s)=\frac{1}{\alpha s+1}\cdot\frac{\tau s+1}{K_0} \qquad (3)$$

  - Response/optimality can be directly deduced by construction of $F_Q(s)$
  - $G^i(s)$, pseudo-inverse of the nominal plant G(s)

$$\rightarrow T_0(s)=\frac{1}{\alpha s+1}$$

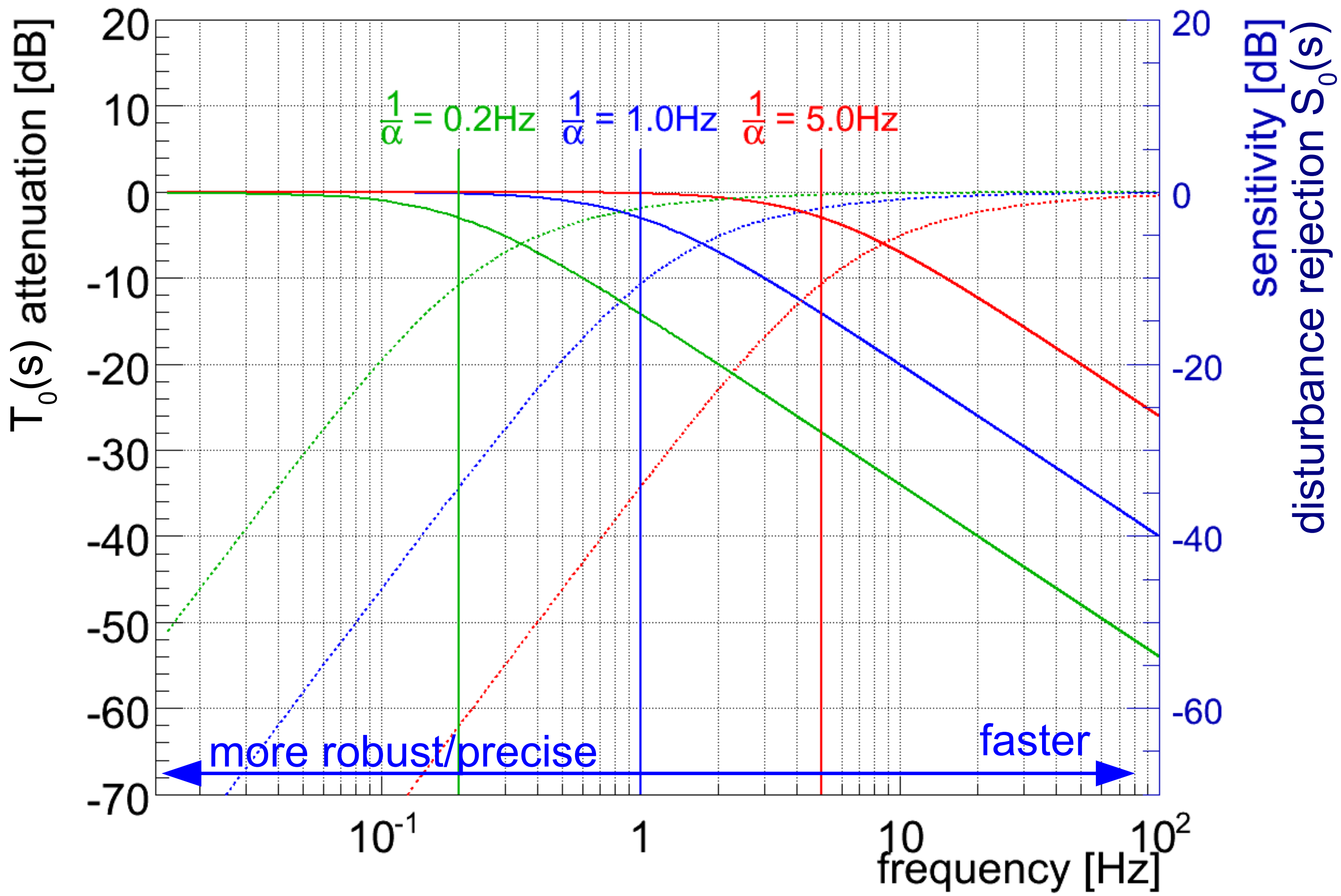
- (1)+(2)+(3) yields the following PI controller:

$$D(s)=K_P+K_i\frac{1}{s} \quad \text{with} \quad K_p=K_0\frac{\tau}{\alpha} \wedge K_i=K_0\frac{1}{\alpha}$$

[1]D. C. Youla et al., *“Modern Wiener-Hopf Design of Optimal Controllers”*, IEEE Trans. on Automatic Control,1976, vol. 21-1,pp. 3-13 & 319-338

$$D(s) = \frac{Q(s)}{1 - Q(s)G(s)}$$

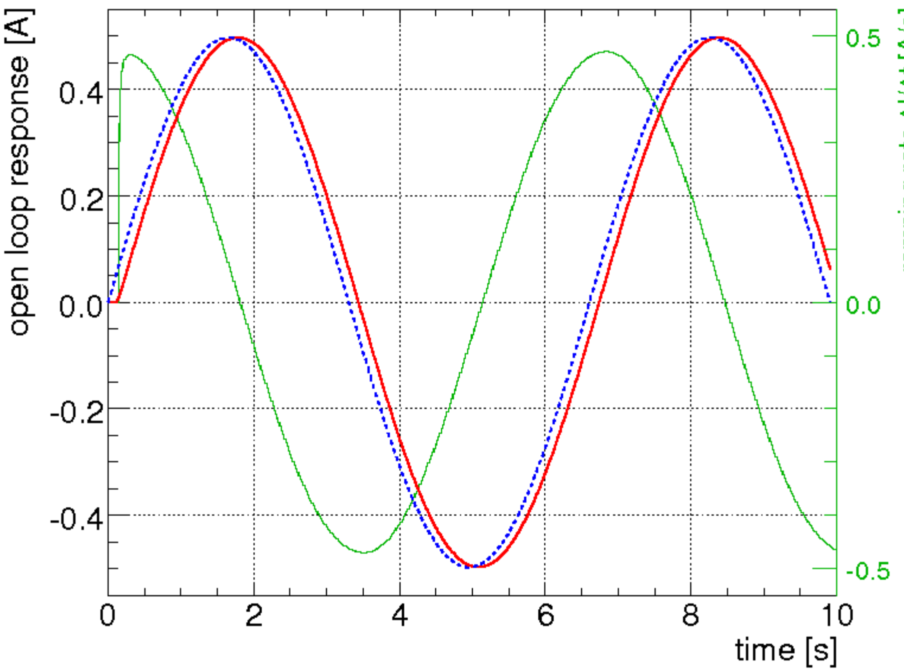
- α > τ…∞ facilitates the trade-off between speed and robustness
  - operator has to deal with one parameter → enables simple adaptive gain-scheduling based on the operational scenario!



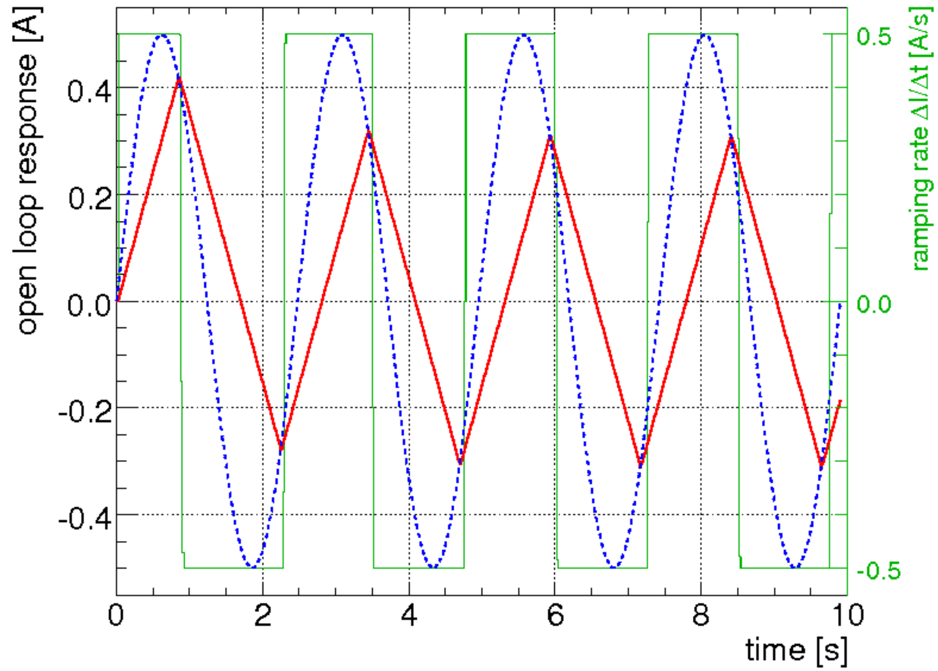
Two common non-linear effects in accelerators:

- Delays: computation, data transmission, dead-time, etc.
- Rate-Limiter: limited slew rate of corrector circuits (due to voltage limitations)
  - e.g. LHC: ±60A converter: $\Delta I/\Delta t|_{max} < 0.5$ A/s

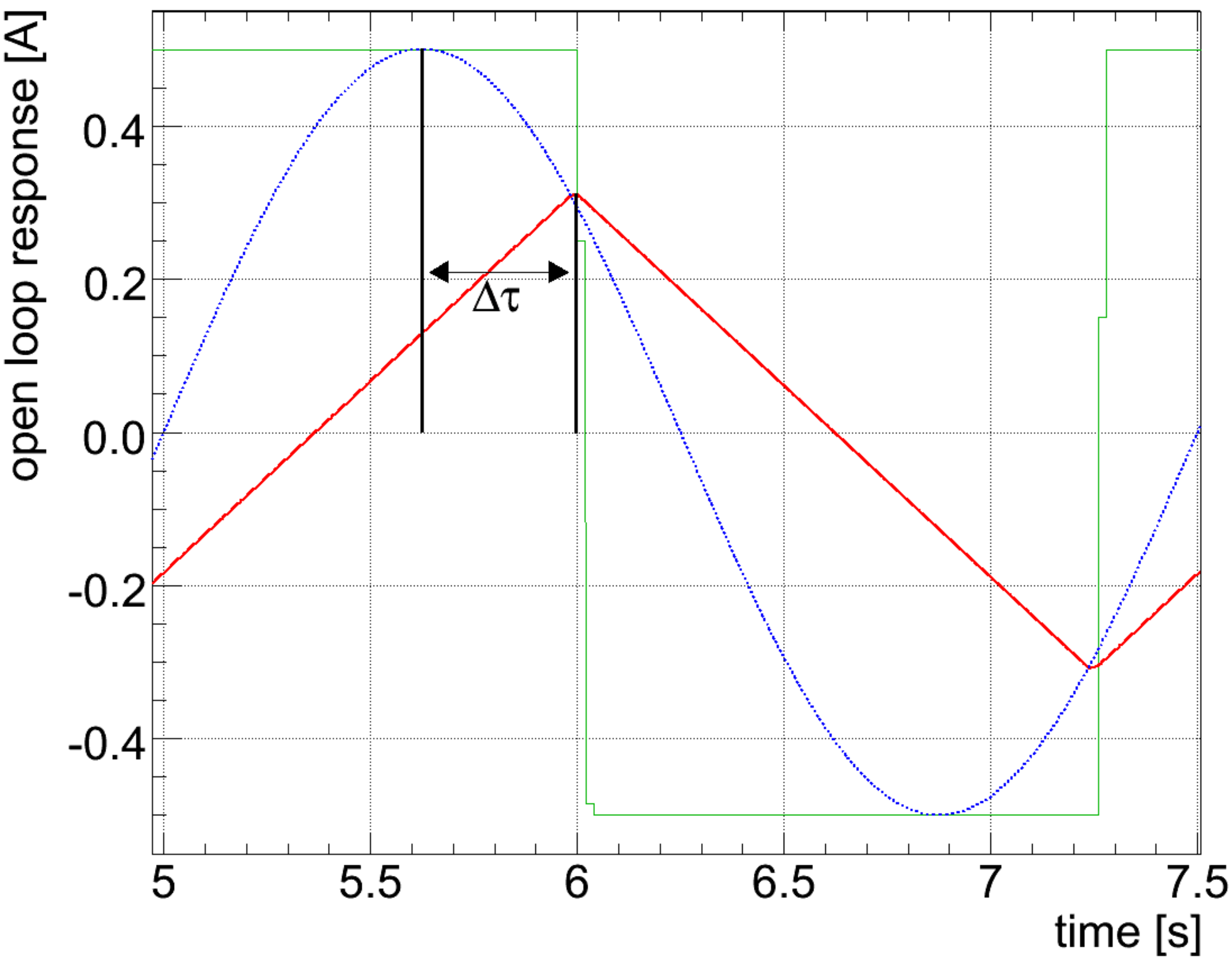
slow perturbation: perfect tracking



fast perturbation: saw-tooth

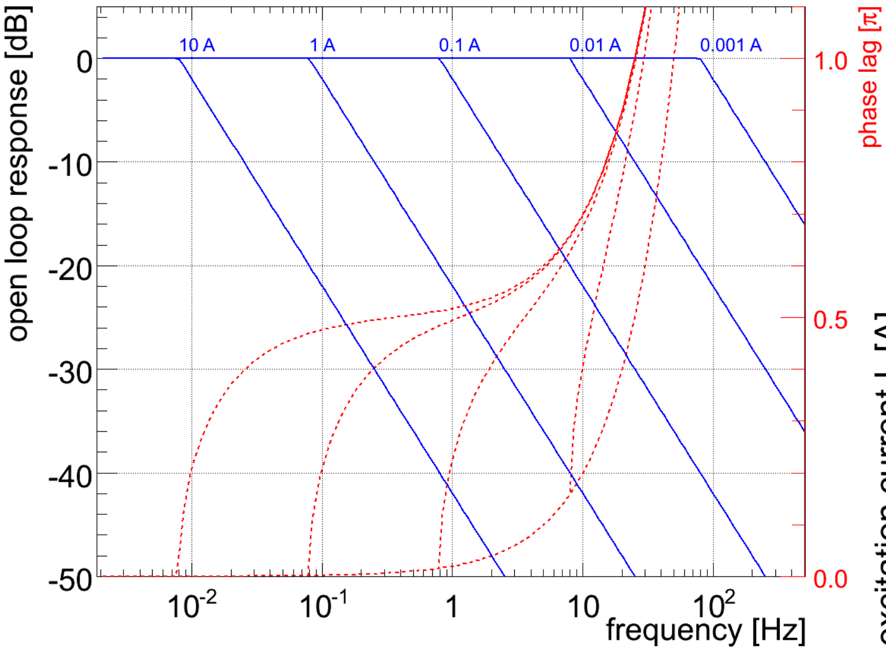


- Rate-limiter in a nut-shell:
  - additional time-delay Δτ that depends on the signal/noise amplitude
  - (secondary: introduces harmonic distortions)

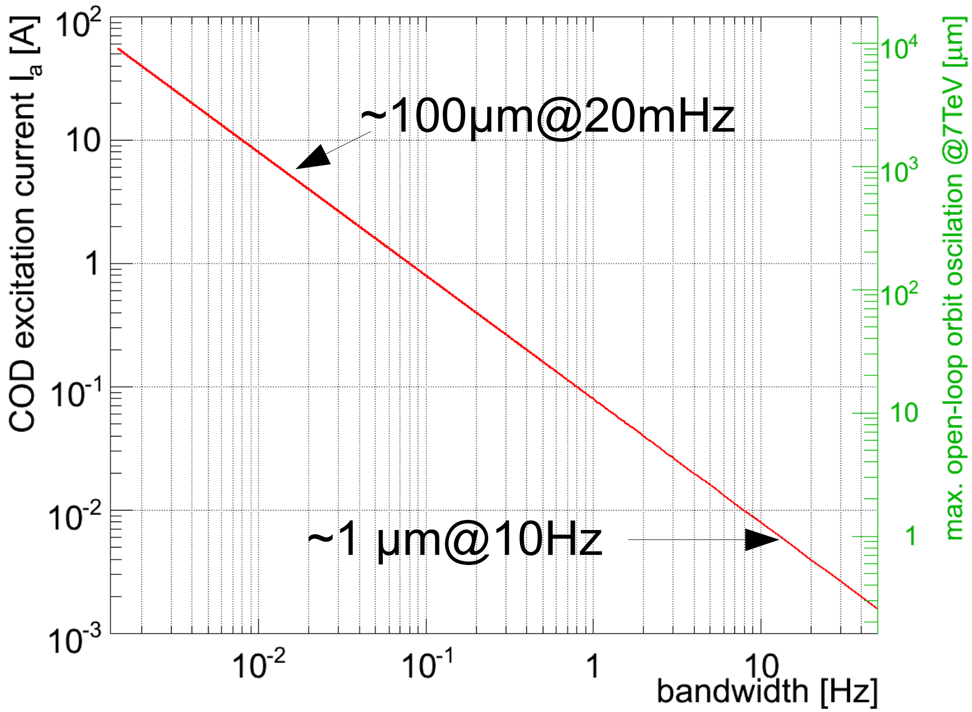


Time-Domain: Non-Linearities III/IV

- Open-loop circuit bandwidth depends on the excitation amplitude:
  - + non-linear phase once rate-limiter is in action...



ΔI=0.1A ↔ Δx≈16 μm@β=180m



- Consider ~16μm@1Hz as effective bandwidth @ 7TeV

Unstable Zeros/non-linearities and delays

$$D(s)=\frac{Q(s)}{1-Q(s)G(s)}$$

- ... cannot a priori be compensated.
  - however, their deteriorating effect on the loop response can be mitigated by taking them into account during the controller design.
- Example: process can be split into stable and instable 'zeros'/components

$$G(s)=\frac{A_0(s)A_u(s)}{B(s)}=G_0(s)\cdot G_{NL}(s)\quad e.g.\quad G(s)=G_0(s)\cdot\underbrace{e^{-\lambda s}}_{\lambda\text{: delay}}$$

- Using the modified ansatz ($F_Q(s)$: desired closed-loop transfer function):

$$Q(s)=F_Q(s)\cdot G^i(s)=F_Q(s)\cdot G_0^{-1}(s)$$

- yields the following closed loop transfer function

$$\rightarrow\quad T(s)\;=Q(s)G(s)=F_Q(s)\cdot G_{NL}(s)\underbrace{=}_{here:}F_Q(s)\cdot e^{-\lambda s}$$

  - Controller design $F_Q(s)$ carried out as for the linear plant
  - Yields known classic predictor schemes:
    - delay → Smith Predictor
    - rate-limit → Anti-Windup Predictor

Example: LHC Feedbacks & Delays + Rate-limiter

$$D(s)=\frac{Q(s)}{1-Q(s)G(s)}$$

- If G(s) contains e.g. delay λ & non-linearities $G_{NL}(s)$

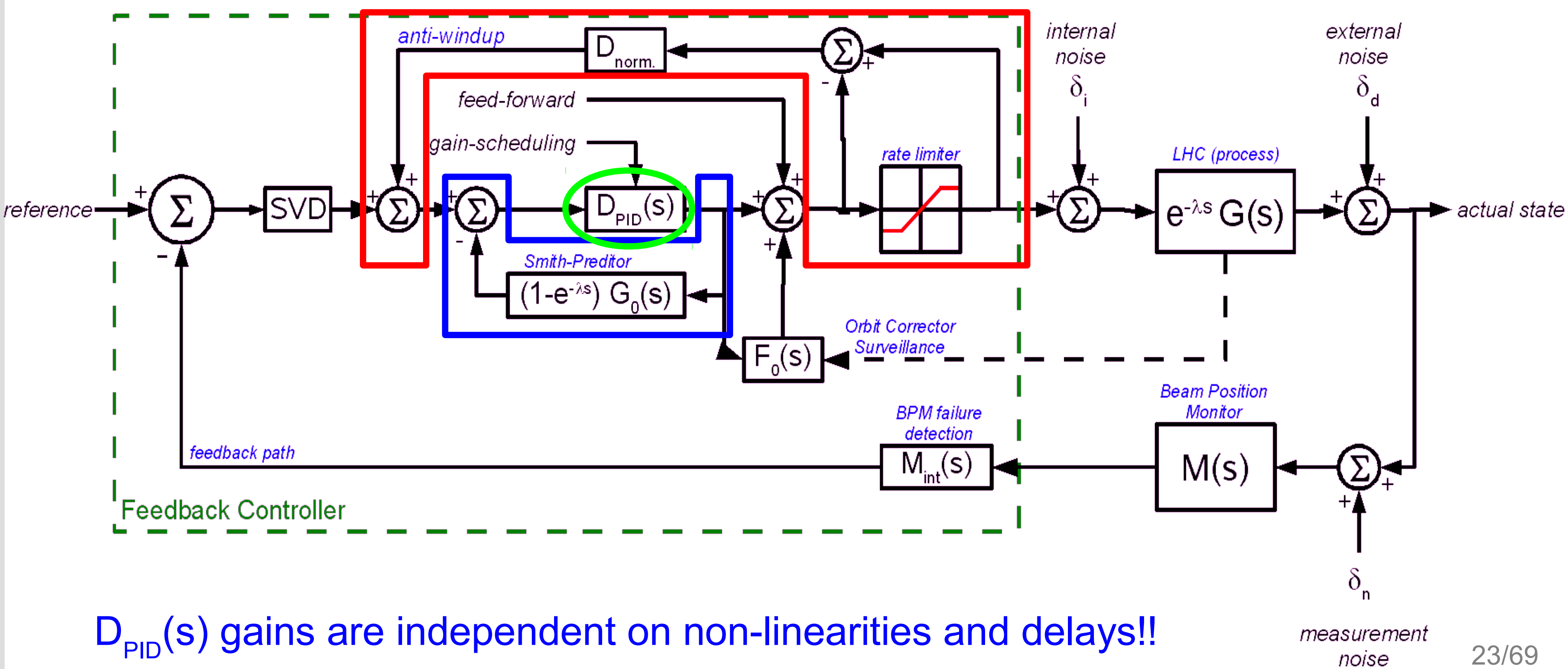
$$G(s)=\frac{e^{-\lambda s}}{\tau s+1}G_{NL}(s)$$

- with τ the power converter time constant and

$$G^{i}(s)=\frac{\tau s+1}{1}$$

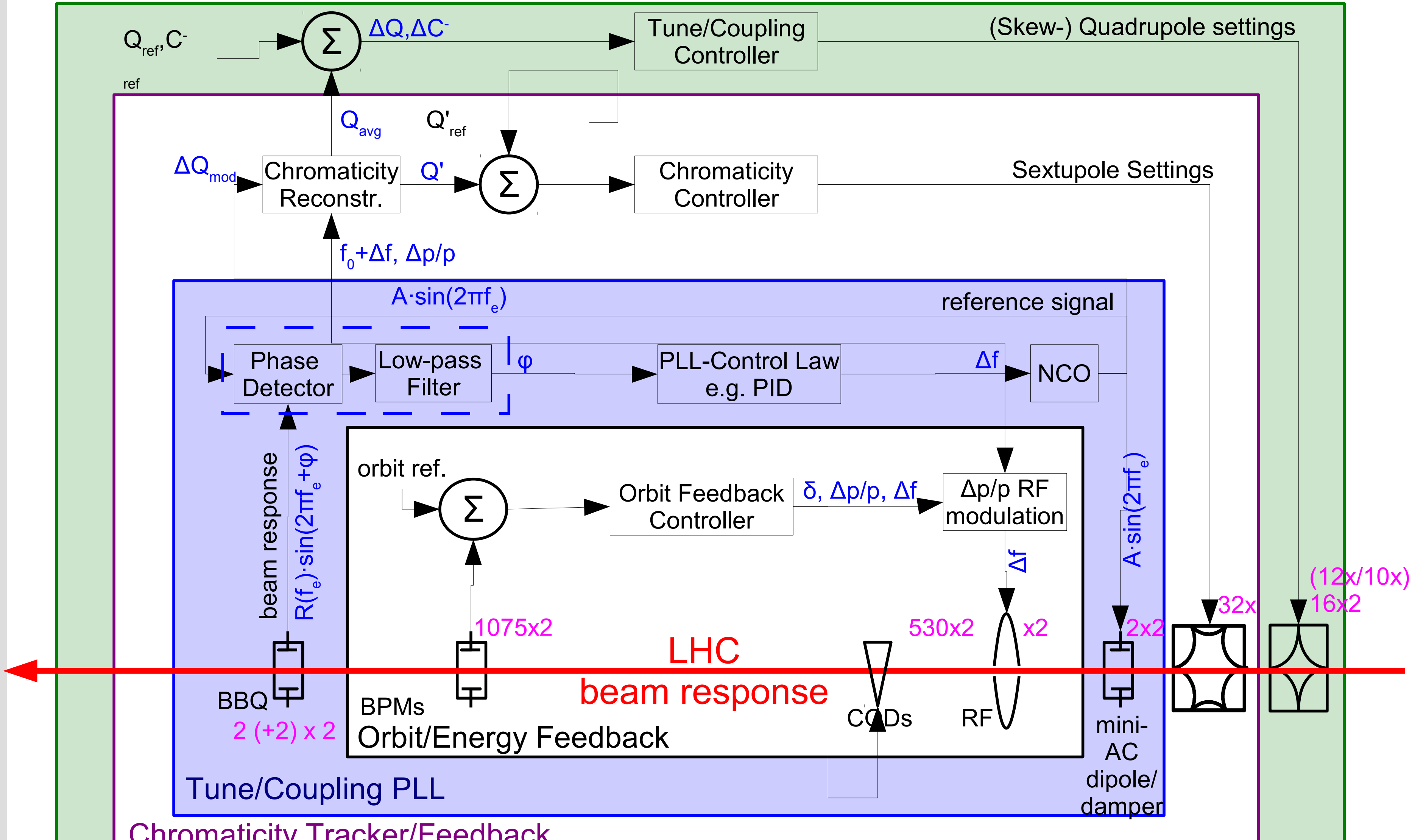
- yields Smith-Predictor and Anti-Windup paths:

$$T(s)=F_{Q}(s)\cdot e^{-\lambda s}G_{NL}(s)$$



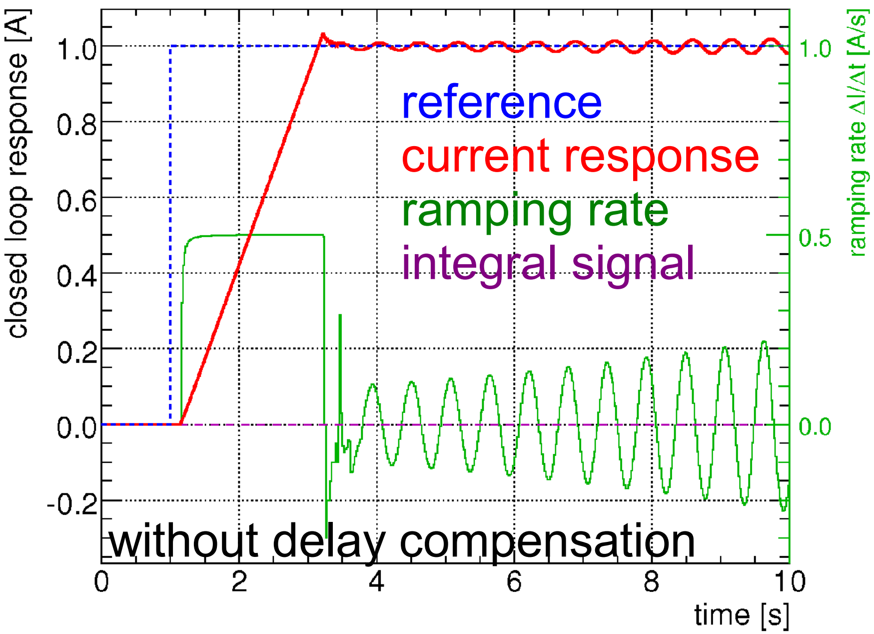
$D_{PID}(s)$ gains are independent on non-linearities and delays!!

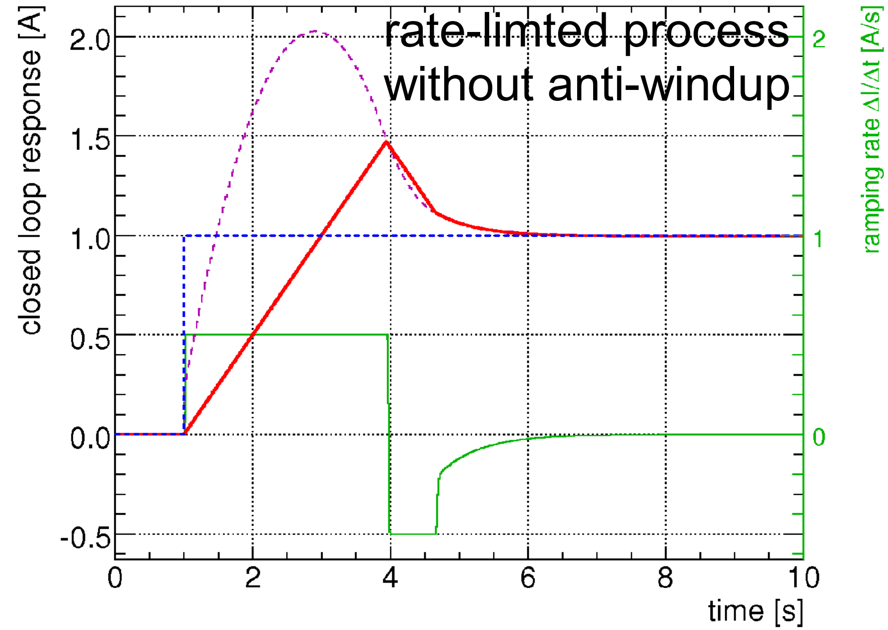
...Conquer:
Cascading between individual Feedbacks



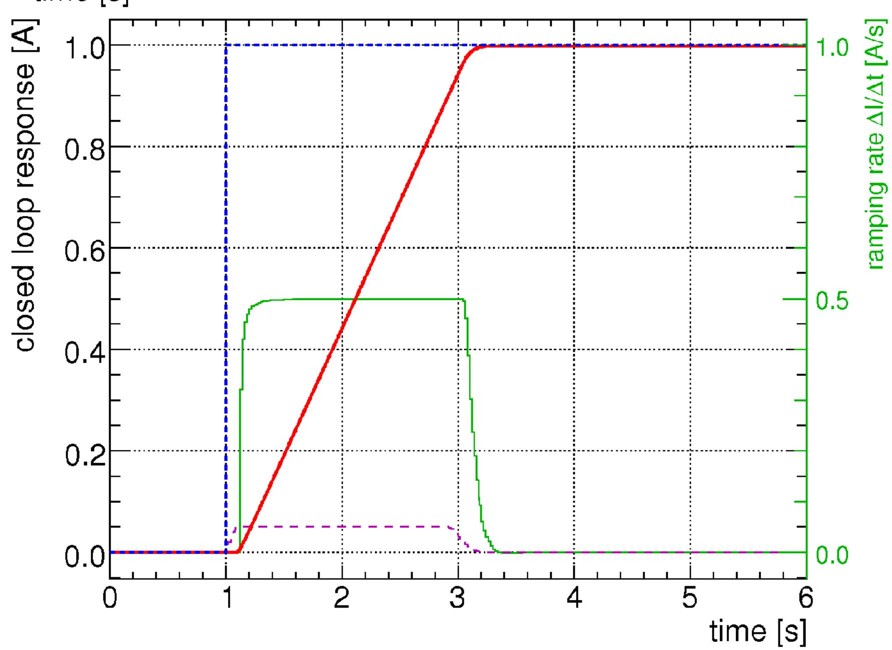
further: $f_{BW}(PLL) \gg f_{BW}(Q') \geq f_{BW}(Q, C^-)$

Motivation for Delay and Rate-Limiter Compensation Example: LHC orbit (Q,Q',C⁻, ...) feedback control





with full delay and windup compensator scheme:



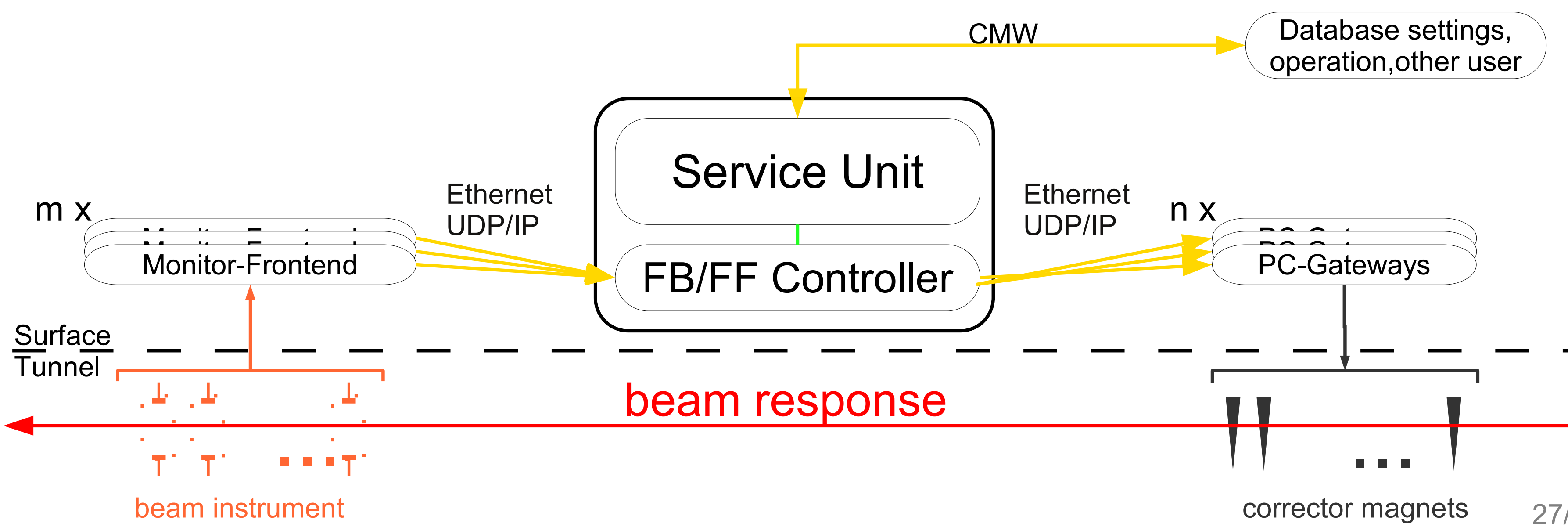
Feedback Sub-Projects: What they do and where to find them...

- Adopted CO-naming convention, common build style deployment
  - Java well integrated but C++ related part still in progress ...
- In svn+ssh://svn.cern.ch/reps/acc-co/lhc/lhc-feedbacks/
  - lhc-app-orbit-feedback-controller – *the actual feedback controller (aka. OFC)*
  - lhc-lib-feedback-commonalities – glue between various OFC parts and OFSU
    - initially separate feedback controller planned → turned out that this is not possible/recommendable but kept stuff in library to minimise profilling and debugging overhead (rarely changes)
  - lhc-lib-twissoptics – physics/optics related code, not FB dependence per se
  - lhc-lib-twissoptics-examples – examples, documentation and unit-type tests
  - lhc-orbitfeedback – *the OFC/OFSU graphical expert user interface*
  - lhc-app-[orbit/tune]-feedback-serviceunit -- *an orphan FESA class*
  - lhc-orbitfeedback-datamanager -- *reference orbit/sequencer (Kajetan)*
  - lhc-orbitfeedback-services -- *reference orbit/sequencer (Kajetan)*
  - optics-server – *LSA-OFSU link to transfer machine optics data (MAD-X style)*
- two noteworthy exceptions – *Orbit, Q/Q' related GUI (aimed at OP usage)*:
  - svn+ssh://svn.cern.ch/reps/acc-co/lhc/lhc-biqp-fixdisplay/
  - svn+ssh://svn.cern.ch/reps/acc-co/accsoft/tuneviewer

Common Feedback/Feed-forward Control Layout

LHC feedback control scheme implementation split into two sub-systems:

- Feedback Controller: actual parameter/feedback controller logic
  - Simple streaming task for all feed-forwards/feedbacks:
    (Monitor → Network $)_{FB}$→ Data-processing → Network → PC-Gateways
  - real-time operating system, constant load, can run auto-triggered
  - Initially targeted to be on an FPGA for reliability reasons
- Service Unit: Interface to users/software control system

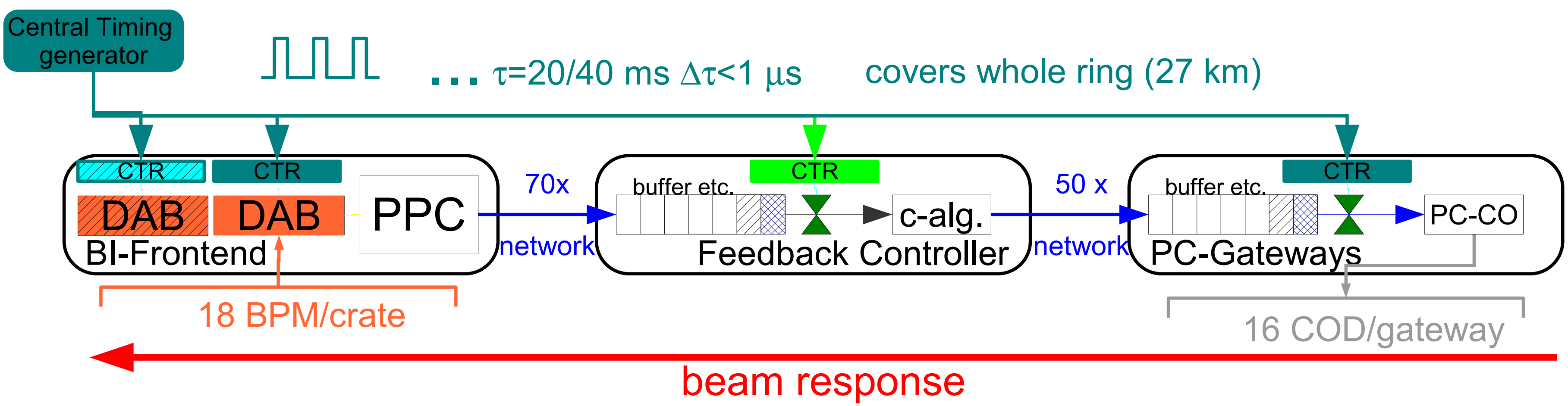


Remaining Jitter Compensation: Fix Max Loop Delay

Two main strategies:

- actual delay measurement and dynamic compensation in SP-branch:
  - only feasible for small systems

- Jitter compensation using a periodic external signal:
  - CERN wide synchronisation of events on sub ms scale
  - The total jitter, the sum of all worst case delays, must stay within “budget”.
  - Measured and anticipated delays and their jitter are well below 20 ms.
  - feedback loop frequency of 50 Hz feasible for LHC, if required...



- Single CTR in OFC == single point of failure

  → dropped it in favour of retrieving timing from multiple BQBBQLHC sources

  → direct UDP software link between BST and OFC for 25 Hz trigger

Why ROOT? A look back on 2004-2005:

- FESA meant LynxOS on modestly performing PCs
  - ~10 ms jitter latency performance (worst: 1-10 s)
  - easily blocked by Ethernet/CMW
  - Limited/no control of locking resource
  - Multi-user environment
    (cannot lock-out user under stress/high load)
  - Leeping real-time constraints was difficult/impossible

→ recognised that time-critical FB business logic needed to be separated from (asynchronuous) user-level requests (GUIs, DB, settings managements, etc.)

- At the same time, needed
  - true real-time latencies in the order of 1-2 ms
  - robust coding standard
    - CO's Java standard was in progress, C++ was bare AB land (and still is)
    - avoid indexing errors, obfuscation of simple linear algebra logic
    - avoid re-implementing the wheel, i.e. numerical tools (fitting)
  - to communicate complex compound structure between various servers

Why ROOT?

- Why to use ROOT framework
  - Widely used platform within/outside HEP
    - several thousand user-base!
    - Supported by CERN staff and other Labs
  - Coding conventions:

Classes begin with **`T`**:	**`TLine, TTree`**
Non-class types end with **`_t`**:	`Int_t`
Data members begin with **`f`**:	`fTree`
Member functions begin with a capital:	`Loop()`
Constants begin with **`k`**:	`kInitialSize, kRed`
Global variables begin with **`g`**:	***`gEnv`***
Static data members begin with **`fg`**:	`fgTokenClient`
Enumeration types begin with **`E`**:	`EColorLevel`
Locals and parameters begin with a lower case:	`nbytes`
Getters and setters begin with **`Get`** and **`Set`**:	`SetLast(), GetFirst()`

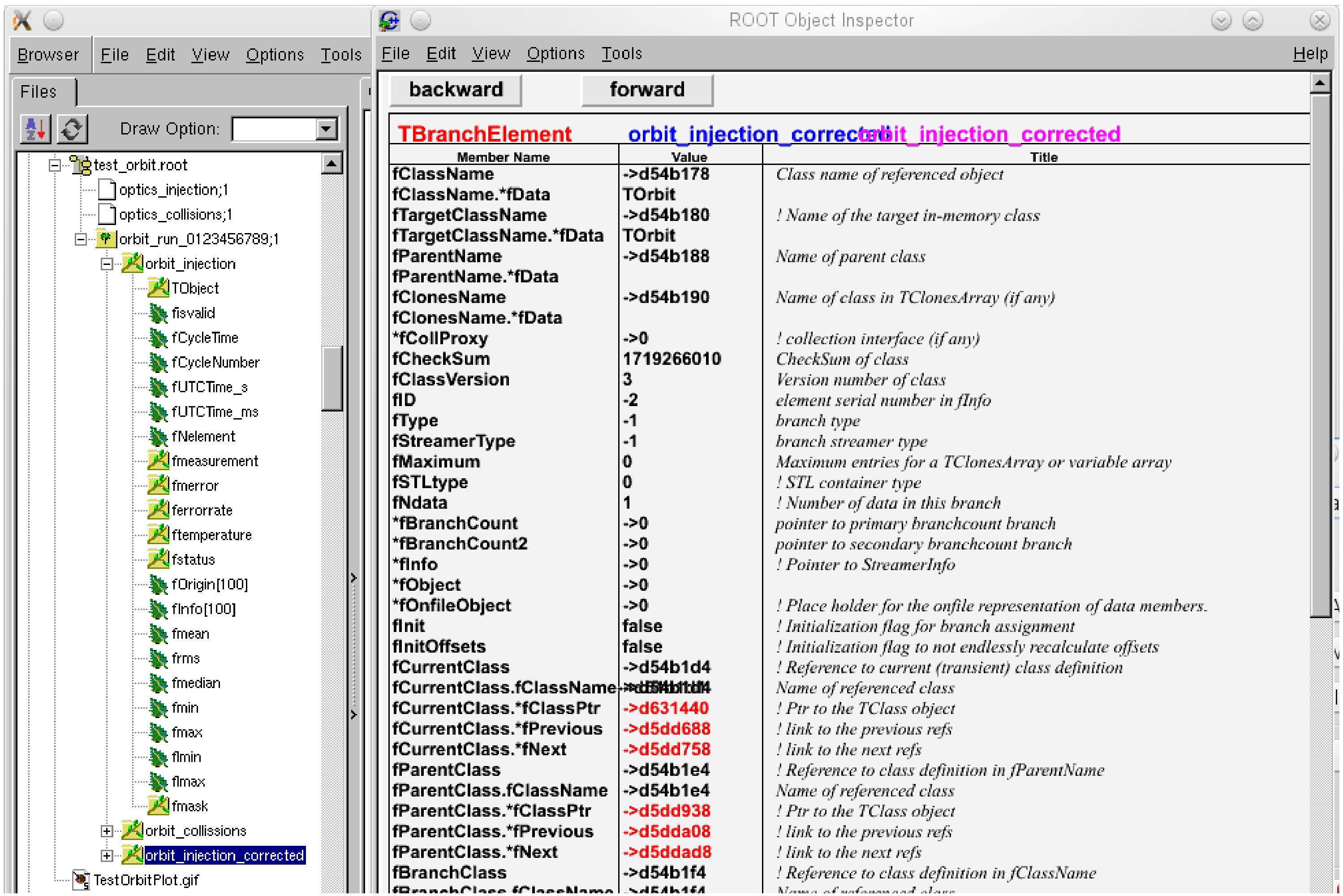
  - Well and actively documented, cross-referenced and checked
    - tutorials, examples, forums, colleagues, ...
  - Accelerated prototyping
    - shell-like development ↔ gcc-style programming possible (CINT)

Why ROOT?

- The OFC code is self-contained and depends only 'gcc' and ROOT
  - Optional: replace in-built libraries with more performant version while keeping the same interface (e.g. FFTW, gsl, …)
  - However: deployment of ROOT/C++ libraries is still at its infancy in CO
- What is specifically used:
  - linear algebra package
    - FB mathematics is encapsulated and described by matrices
    - type, dimension, index safety!
  - True Chi² fitting – numerically tested no 'hack' solution
  - Most OFC data are complex structures composed of scalar, vector, string, lists, …, data that need to be synchronised and
    - Internally copied
    - Communicated to the OFSU
    - Efficiently written to file
  - Case-/User-specific code possible but with very high risk of obfuscation, consistency errors and omission of data copy routines, etc...

Why ROOT? I/O streamers and CINT

- Objects derived from 'TObject' allow automated streamer function generation 'void Streamer(TBuffer& b)' that allows to convert complex object structures into linear arrays that can be efficiently copied, transmitted or written to file.
  - independent of '32 vs 64', 'big-vs.little endian', ROOT version, ...



- TInterlink implements a basic RPC with streaming data from/to OFC/OFSU
- Registered functions such as:

```
interlink->RegisterFunction(this, (TObjectFunction)&OrbitCorrection::SetOrbitFBStateH,
    "OrbitFBStateH", kWRITE, TCallBack::kNONE, TCallBack::kBool_t,
    "sets horizontal OrbitFB state: kTRUE -> on, kFALSE -> off []");

interlink->RegisterFunction(this, (TObjectFunction)&OrbitCorrection::GetOrbitFBStateH,
    "OrbitFBStateH", kREAD, TCallBack::kBool_t, TCallBack::kNONE,
    "returns horizontal OrbitFB state: kTRUE -> on, kFALSE -> off []");

interlink->RegisterFunction(this, (TObjectFunction)&OrbitCorrection::GetOrbitDifferenceH,
    "OrbitDifferenceH", kREAD, TCallBack::kTObject, TCallBack::kNONE,
    "horizontal orbit difference to reference orbit [TOrbit]");
```

- Can be remotely invoked via:
  - “get OrbitFBStateH“ or “set OrbitFBStateH true“
  - 'get OrbitDifferenceH' with return being a serialised TOrbit object
- Important, the list of all available OFC commands can be retrieved via “get commands”

I/O streamers and CINT → Tinterlink II/II

- A total of 554 commands (~half a 'get' the other 'set'):
  mostly simple scalar commands like 'switch OFB on/off', gains, …

```
16: get IOMessageRawIndex     <Int_t>       info: <TInterlink> returns the message at the raw index of the console IO's circular buffer [TObjString] (returns <TObject>)
17: set ShutDownRequest       <Bool_t>      info: <TInterlink> kTRUE: request a shut-down of TInterlink and associated processes (returns <null>)
18: get UpTime                <null>        info: <TInterlink> returns TInterlink's up-time in seconds (returns <Long_t>)
19: get NumberCPUs            <null>        info: <TInterlink> returns number of available CPUs/cores (returns <Int_t>)
20: get CPUID                 <null>        info: <TInterlink> returns CPU ID the TInterlink interface is running on (returns <Int_t>)
21: get CPUClockFrequency     <null>        info: <TInterlink> returns CPU clock frequency [GHz] (returns <Double_t>)
22: get IsRTLinuxKernel       <null>        info: <TInterlink> returns wether TInterlink runs on a RT linux system (returns <Bool_t>)
23: set ResetRTLatencies      <null>        info: <TInterlink> resets kernel RT latency histograms (returns <Bool_t>)
24: get MaxKernelLatency      <null>        info: <TInterlink> returns maximum CPU wake-up latency (returns <Int_t>)
25: get KernelLatencyHistogram  <Int_t>     info: <TInterlink> returns maximum CPU wake-up latency histogram for given CPU [TH1F] (returns <TObject>)
26: set UseEnergyOFC          <Bool_t>      info: <MachineState> true: use energy from OFC's timing telegrams [] (returns <null>)
27: get UseEnergyOFC          <null>        info: <MachineState> true: uses energy from OFC's timing telegrams [Bool_t] (returns <Bool_t>)
28: set forceBeamPresence     <Bool_t>      info: <MachineState> true: forces beam presence flags to true (returns <null>)
29: get forceBeamPresence     <null>        info: <MachineState> true: beam presence flag is forced to true (returns <Bool_t>)
30: set EnergyRefOFSU         <Double_t>    info: <MachineState> energy reference (OFSU interface) [GeV] (returns <null>)
31: get EnergyRefOFSU         <null>        info: <MachineState> energy reference in [GeV] (OFSU interface) [Double_t] (returns <Double_t>)
32: get EnergyRefOFC          <null>        info: <MachineState> energy reference in [GeV] (OFC interface) [Double_t] (returns <Double_t>)
33: get Energy                <null>        info: <MachineState> energy reference in GeV [Double_t] (returns <Double_t>)
34: get FillNumber            <null>        info: <MachineState> the magical ever increasing LHC fill number (returns <Int_t>)
35: get BeamIntensityB1       <null>        info: <MachineState> beam 1 intensity in protons/beam (returns <Double_t>)
36: get BeamIntensityB2       <null>        info: <MachineState> beam 2 intensity in protons/beam (returns <Double_t>)
37: get BunchIntensityB1      <null>        info: <MachineState> beam 1 avg. bunch intensity in protons/bunch (returns <Double_t>)
38: get BunchIntensityB2      <null>        info: <MachineState> beam 2 avg. bunch intensity in protons/bunch (returns <Double_t>)
39: get NumberOfBunchesB1     <null>        info: <MachineState> number of bunches in beam 1 (returns <Int_t>)
40: get NumberOfBunchesB2     <null>        info: <MachineState> number of bunches in beam 2 (returns <Int_t>)
41: get BunchSpacingB1        <null>        info: <MachineState> bunch spacing B1 [ns] (returns <Double_t>)
42: get BunchSpacingB2        <null>        info: <MachineState> bunch spacing B2 [ns] (returns <Double_t>)
43: get BeamPresentFlagB1     <null>        info: <MachineState> beam present flag B1 (returns <Bool_t>)
44: get BeamPresentFlagB2     <null>        info: <MachineState> beam present flag B2 (returns <Bool_t>)
45: get SetupBeamFlagB1       <null>        info: <MachineState> setup beam flag B1 (returns <Bool_t>)
46: get SetupBeamFlagB2       <null>        info: <MachineState> setup beam flag B2 (returns <Bool_t>)
47: get StableBeamFlagB1      <null>        info: <MachineState> stable beam flag B1 (returns <Bool_t>)
48: get StableBeamFlagB2      <null>        info: <MachineState> stable beam flag B2 (returns <Bool_t>)
49: get MovableDevicesAllowedFlagB1  <null>        info: <MachineState> movable devices allowed flag B1 (returns <Bool_t>)
50: get MovableDevicesAllowedFlagB2  <null>        info: <MachineState> movable devices allowed flag B2 (returns <Bool_t>)
51: get BeamMode              <null>        info: <MachineState> LHC Beam Mode raw enumeration (prototype) (returns <Int_t>)
52: set ResetMachineState     <null>        info: <MachineState> resets LHC machine state as tracked by the OFC (returns <null>)
53: get IsOperationalServer   <null>        info: <MachineState> true: operational, false: no (returns <Bool_t>)
54: set RTMasterSwitch        <Bool_t>      info: <MachineState> true: FB is allowed to send RT trims (returns <null>)
55: get RTMasterSwitch        <null>        info: <MachineState> true: FB is allowed to send RT trims (returns <Bool_t>)
56: set SystemValidated       <Bool_t>      info: <MachineState> true: system has been checked after a reboot or crash (returns <null>)
57: get SystemValidated       <null>        info: <MachineState> true: system has been checked after a reboot or crash (returns <Bool_t>)
58: get SystemStartTime       <null>        info: <MachineState> UTC seconds since 1970 when system was started (returns <Double_t>)
59: get SystemValidationTime  <null>        info: <MachineState> UTC seconds since 1970 when system was validates (returns <Double_t>)
60: set ParticleTypeB1        <Int_t>       info: <MachineState> particle type of beam 1 [1: proton: 2, Pb82: 3: PTYPE_AR18, ...] (returns <null>)
61: set ParticleTypeB2        <Int_t>       info: <MachineState> particle type of beam 2 [1: proton: 2, Pb82: 3: PTYPE_AR18, ...] (returns <null>)
62: get ParticleTypeB1        <null>        info: <MachineState> particle type of beam 1 [1: proton: 2, Pb82: 3: PTYPE_AR18, ...] (returns <Int_t>)
63: get ParticleTypeB2        <null>        info: <MachineState> particle type of beam 2 [1: proton: 2, Pb82: 3: PTYPE_AR18, ...] (returns <Int_t>)
64: get ChargeMassRatioB1     <null>        info: <MachineState> B1 particle charge-to-mass ratio [] (returns <Double_t>)
65: get ChargeMassRatioB2     <null>        info: <MachineState> B2 particle charge-to-mass ratio [] (returns <Double_t>)
66: get OpticH                <Int_t>       info: <ReferenceOptics> returns horziontal response matrix for given slot number [TResponseMatrix] (returns <TObject>)
67: get OpticV                <Int_t>       info: <ReferenceOptics> returns vertical response matrix for given slot number [TResponseMatrix] (returns <TObject>)
68: get OpticNameH            <Int_t>       info: <ReferenceOptics> returns horziontal optic name for given slot number [TObjString] (returns <TObject>)
69: get OpticNameV            <Int_t>       info: <ReferenceOptics> returns vertical optic name for given slot number [TObjString] (returns <TObject>)
70: get MaxNOptics            <null>        info: <ReferenceOptics> returns maximum number of available optic slots (returns <Int_t>)
```

- Important: provides not only list and short description but also location (object) where the specific command is implemented

General Orbit Feedback Controller Structure I/II

- Main streaming taks contained in 'OFBController.cpp', logic flow:
- <general initialisation>
- Main Loop
  - Data accumulation loop (free-running or locked at 25 Hz):
    - BPMConcentrator – *nomen est omen*
    - QQPConcentrator, MachineState – *nomen est omen*
  - <validate setting and received data>
  - <update references>
  - EnergyCorrection – radial loop feedback, radial modulation, …
  - OrbitCorrection – *orbit feedback space domain*
    - Wakes up two worker threads performing the two $O(n^2)$ multiplication
  - QQPConcentrator – *tune feedback space and time domain*
  - *<send COD and Q/Q' corrector data>*
  - *<publish/stream OFC state via UDP to OFSU>*
  - *<wait up to 5 ms or for remainder of iteration, service TInterlink requests>*
- <general de-initialisation/restart>

General Orbit Feedback Controller Structure II/II

- Additional independent tasks/threads:
  - Tinterlink – *RPC class executed only once the main task is finished*
    - blocked most of the time, except at the end of very main iteration
  - CODConcentrator – *FGC data concentrator*
    - free running/constant load → long-term: synchronise to BPMs' 25 Hz rate
  - ReferenceOpticsMagic – *OFC-based optics recomputation*
    - High CPU load and risk of stalling the OFC (was put there initially as a hack)→ should be migrated to OFSU
- Normal 'top' load on cs-ccr-ofc:

```
top - 00:32:56 up 261 days,  9:38,  1 user,  load average: 0.97, 0.78, 0.75
Tasks: 158 total,   2 running, 156 sleeping,   0 stopped,   0 zombie
Cpu(s): 16.3%us,  1.2%sy,  0.0%ni, 81.4%id,  0.1%wa,  0.1%hi,  1.0%si,  0.0%st
Mem:   4148320k total,  3385196k used,   763124k free,   441636k buffers
Swap:  5421928k total,        0k used,  5421928k free,   884900k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 9565 root      -3   0 1414m 1.4g  41m R 51.5 34.9  19738:06 OFBController
 9566 root     -12   0 1414m 1.4g  41m S  7.0 34.9   2659:16 OFBController
 9567 root     -12   0 1414m 1.4g  41m S  6.3 34.9   2347:34 OFBController
 9570 root     -34   0 1414m 1.4g  41m S  2.0 34.9 572:46.38 OFBController
 9571 root     -34   0 1414m 1.4g  41m S  1.7 34.9 629:28.38 OFBController
    4 root     -71  -5     0    0    0 S  0.3  0.0  20:50.57 ksoftirqd/0
 3287 root      39  19     0    0    0 S  0.3  0.0 994:18.85 kipmi0
```

Main loop
Orbit-FB-H
Orbit-FB-V
TInterlink
CODConcentrator

Network Traffic In and Out I/II

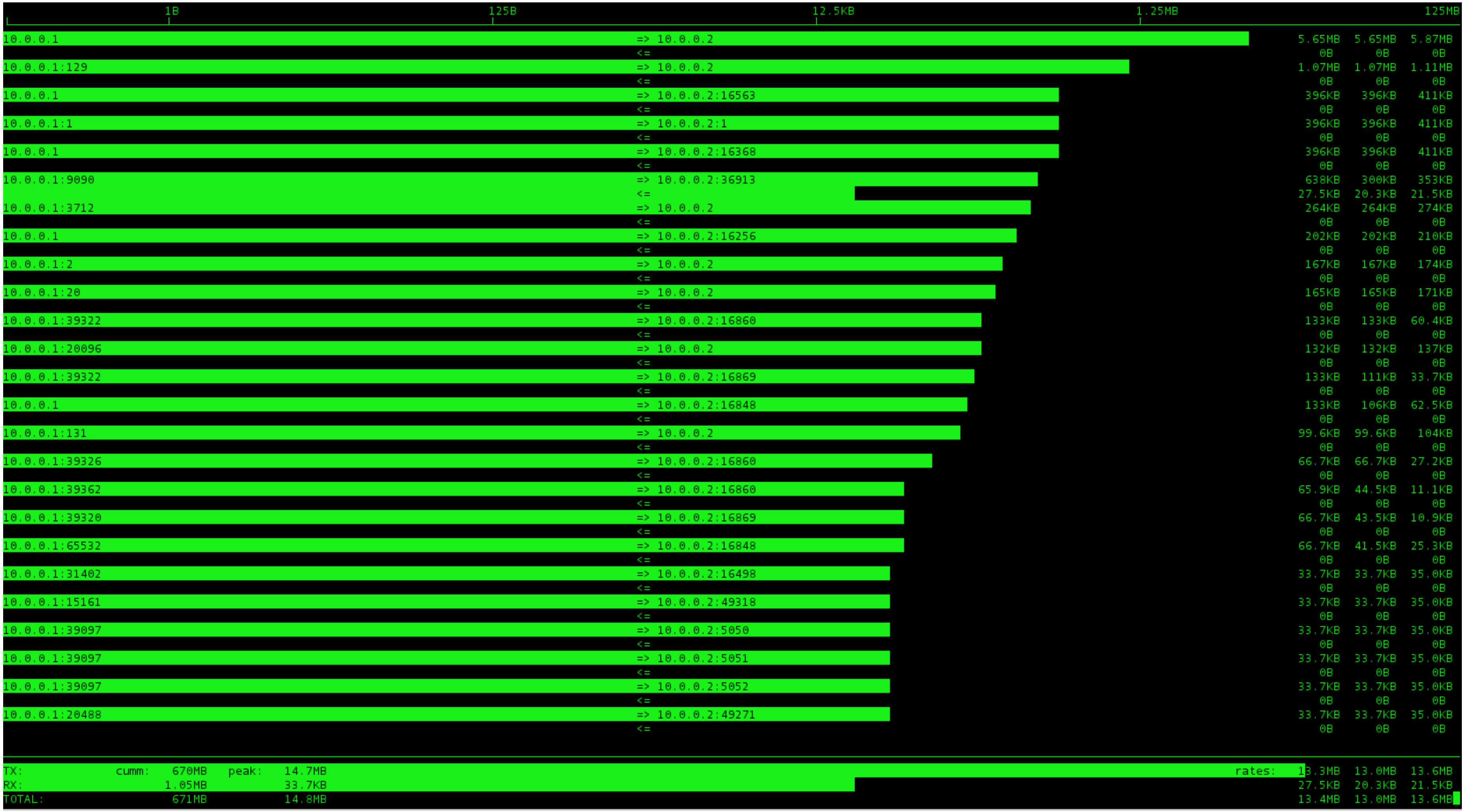
- '/usr/sbin/iftop' is your friend, typical output on cs-ccr-ofc:

```
                     10b               100b              1.00Kb            10.0Kb            100Kb             1.00Mb            10.0Mb          100Mb
cs-ccr-ofc.cern.ch                                         <=> cfv-sr3-bpmb2ra.cern.ch                                  220Kb  220Kb  220Kb
cs-ccr-ofc.cern.ch                                         <=> 172.18.28.96                                             220Kb  220Kb  220Kb
cs-ccr-ofc.cern.ch                                         <=> cfv-sr5-bpmb1lb.cern.ch                                  220Kb  220Kb  220Kb
cs-ccr-ofc.cern.ch                                         <=> 172.18.29.13                                             220Kb  220Kb  220Kb
cs-ccr-ofc.cern.ch                                         <=> 172.18.25.151                                            220Kb  220Kb  220Kb
cs-ccr-ofc.cern.ch                                         <=> 172.18.58.252                                            220Kb  220Kb  220Kb
cs-ccr-ofc.cern.ch                                         <=> 172.18.58.253                                            220Kb  220Kb  220Kb
cs-ccr-ofc.cern.ch                                         <=> 172.18.50.187                                            220Kb  220Kb  220Kb
cs-ccr-ofc.cern.ch                                         <=> 172.18.25.157                                            220Kb  220Kb  220Kb
cs-ccr-ofc.cern.ch                                         <=> 172.18.58.254                                            220Kb  220Kb  220Kb
cs-ccr-ofc.cern.ch                                         <=> 172.18.29.16                                             220Kb  220Kb  220Kb
cs-ccr-ofc.cern.ch                                         <=> cfv-sx4-bpmb1rb.cern.ch                                  220Kb  220Kb  220Kb
cs-ccr-ofc.cern.ch                                         <=> cfv-sr6-bpmint2.cern.ch                                  220Kb  220Kb  220Kb
cs-ccr-ofc.cern.ch                                         <=> cfv-sr6-bpmb1lb.cern.ch                                  220Kb  220Kb  220Kb
cs-ccr-ofc.cern.ch                                         <=> 172.18.58.251                                            220Kb  220Kb  220Kb
cs-ccr-ofc.cern.ch                                         <=> cfv-sr6-bpmb1la.cern.ch                                  220Kb  220Kb  220Kb
cs-ccr-ofc.cern.ch                                         <=> cfv-sr5-bpmb1la.cern.ch                                  220Kb  220Kb  220Kb
cs-ccr-ofc.cern.ch                                         <=> cfv-sx4-bpmb1ra.cern.ch                                  220Kb  220Kb  220Kb
cs-ccr-ofc.cern.ch                                         <=> cfv-sr5-bpmb2lt.cern.ch                                  220Kb  220Kb  220Kb
cs-ccr-ofc.cern.ch                                         <=> cfv-sr8-bpmb1rb.cern.ch                                  220Kb  220Kb  220Kb
cs-ccr-ofc.cern.ch                                         <=> 172.18.43.99                                             220Kb  220Kb  220Kb
cs-ccr-ofc.cern.ch                                         <=> 172.18.43.102                                            220Kb  220Kb  220Kb
cs-ccr-ofc.cern.ch                                         <=> cfv-sr5-bpmb1rt.cern.ch                                  220Kb  220Kb  220Kb
cs-ccr-ofc.cern.ch                                         <=> cfv-sr6-bpmint1.cern.ch                                  220Kb  220Kb  220Kb
cs-ccr-ofc.cern.ch                                         <=> cfv-sr2-bpmb2rb.cern.ch                                  220Kb  220Kb  220Kb
cs-ccr-ofc.cern.ch                                         <=> 172.18.60.40                                             220Kb  220Kb  220Kb
cs-ccr-ofc.cern.ch                                         <=> 172.18.43.100                                            220Kb  220Kb  220Kb
cs-ccr-ofc.cern.ch                                         <=> cfv-sr8-bpmb1la.cern.ch                                  220Kb  220Kb  220Kb
cs-ccr-ofc.cern.ch                                         <=> 172.18.60.35                                             220Kb  220Kb  220Kb
cs-ccr-ofc.cern.ch                                         <=> 172.18.26.28                                             220Kb  220Kb  220Kb
cs-ccr-ofc.cern.ch                                         <=> cfv-sr1-bpmb1la.cern.ch                                  220Kb  220Kb  220Kb
cs-ccr-ofc.cern.ch                                         <=> cfv-sr5-bpmb2la.cern.ch                                  220Kb  220Kb  220Kb
cs-ccr-ofc.cern.ch                                         <=> cfv-sr5-bpmb2lb.cern.ch                                  220Kb  220Kb  220Kb
cs-ccr-ofc.cern.ch                                         <=> cfv-sr1-bpme.cern.ch                                     220Kb  220Kb  220Kb
cs-ccr-ofc.cern.ch                                         <=> 172.18.26.27                                             220Kb  220Kb  220Kb
cs-ccr-ofc.cern.ch                                         <=> 172.18.50.184                                            220Kb  220Kb  220Kb
cs-ccr-ofc.cern.ch                                         <=> 172.18.28.92                                             220Kb  220Kb  220Kb
cs-ccr-ofc.cern.ch                                         <=> cfv-sr6-bpmb2lb.cern.ch                                  220Kb  220Kb  220Kb
cs-ccr-ofc.cern.ch                                         <=> cfv-sr6-bpmb2la.cern.ch                                  220Kb  220Kb  220Kb
cs-ccr-ofc.cern.ch                                         <=> cfv-sr6-bpmb1rb.cern.ch                                  220Kb  220Kb  220Kb
cs-ccr-ofc.cern.ch                                         <=> 172.18.60.39                                             220Kb  220Kb  220Kb
cs-ccr-ofc.cern.ch                                         <=> 172.18.26.24                                             220Kb  220Kb  220Kb
cs-ccr-ofc.cern.ch                                         <=> 172.18.29.19                                             220Kb  220Kb  219Kb
cs-ccr-ofc.cern.ch                                         <=> 172.18.28.99                                             220Kb  220Kb  219Kb
cs-ccr-ofc.cern.ch                                         <=> 172.18.25.153                                            220Kb  220Kb  219Kb
cs-ccr-ofc.cern.ch                                         <=> 172.18.29.17                                             220Kb  220Kb  219Kb
cs-ccr-ofc.cern.ch                                         <=> 172.18.50.182                                            220Kb  220Kb  219Kb
cs-ccr-ofc.cern.ch                                         <=> 172.18.50.185                                            220Kb  220Kb  219Kb
cs-ccr-ofc.cern.ch                                         <=> 172.18.29.18                                             220Kb  220Kb  219Kb
cs-ccr-ofc.cern.ch                                         <=> 172.18.60.38                                             220Kb  220Kb  219Kb

TX:             cumm:  57.3MB   peak:   6.58Mb                                                            rates:   6.45Mb  6.45Mb  6.46Mb
RX:                    138MB            15.6Mb                                                                     15.6Mb  15.6Mb  15.6Mb
TOTAL:                 195MB            22.2Mb                                                                     22.1Mb  22.0Mb  22.0Mb
```

- You can scroll up/down with 'k' and 'j', 'L' toggles logarithmic display, 't' toggles in/out traffic display, 'h' for help and advanced port/DNS display
- Healthy state: all BPMs, FGC Gateways send with the same data rate

- Healthy OFC-OFSU communication:



- Alternatively: 'netstat -Natn' and 'netstat -Naun' on cs-ccr-of[c/su] indicate if the network sockets are overloaded (via Recv-Q Send-Q)

IO Messages and how to use them:

- Printing to console is hazardous in an RT environment since it can block the process depending on the state of the serial console
- Instead: implemented a circular buffer which is written to by all OFC, twiss-optics, ROOT, etc function, e.g.:

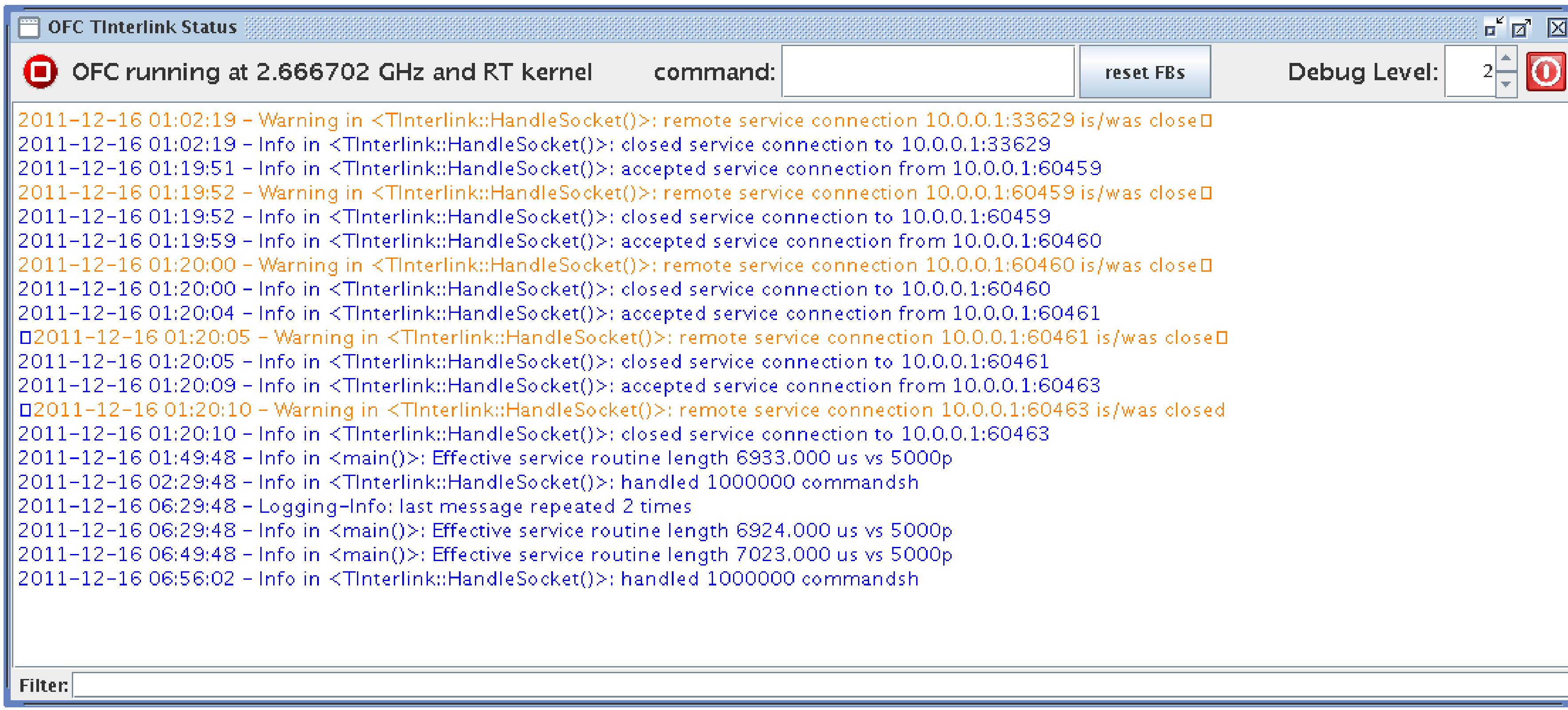
```
2011-12-13 07:05:46 - Error in <BPMConcentrator::CheckDoubleValue(range)>: value +0.000000e+00 at index 10 in dabTemp is out of range [+1.000000e+01, +1.000000e+02]
```

```
2011-12-13 07:05:46 - Error in <BPMConcentrator::CheckDoubleValue(range)>: value +0.000000e+00 at index 10 in dabTemp is out of range [+1.000000e+01, +1.000000e+02]
```

- After quick check in BPMConcentrator.cpp:1559 one finds:
- ```
  [..]
  unsigned short ttemperature_short = SWAP_USHORT(data.dabTemp[i]);
  Double_t ttemperature = CheckDoubleValue(0.1*ttemperature_short, 0.0, tempStatus, i, "dabTemp", 10.0, 100.0); // [10, 100] degC
  [..]
  ```
  ```

IO Messages and how to use them:

- Messages can be monitored via the Orbit-FB GUI and/or BI-QP Fix-Display
  - Would need to be logged for post-mortem analysis



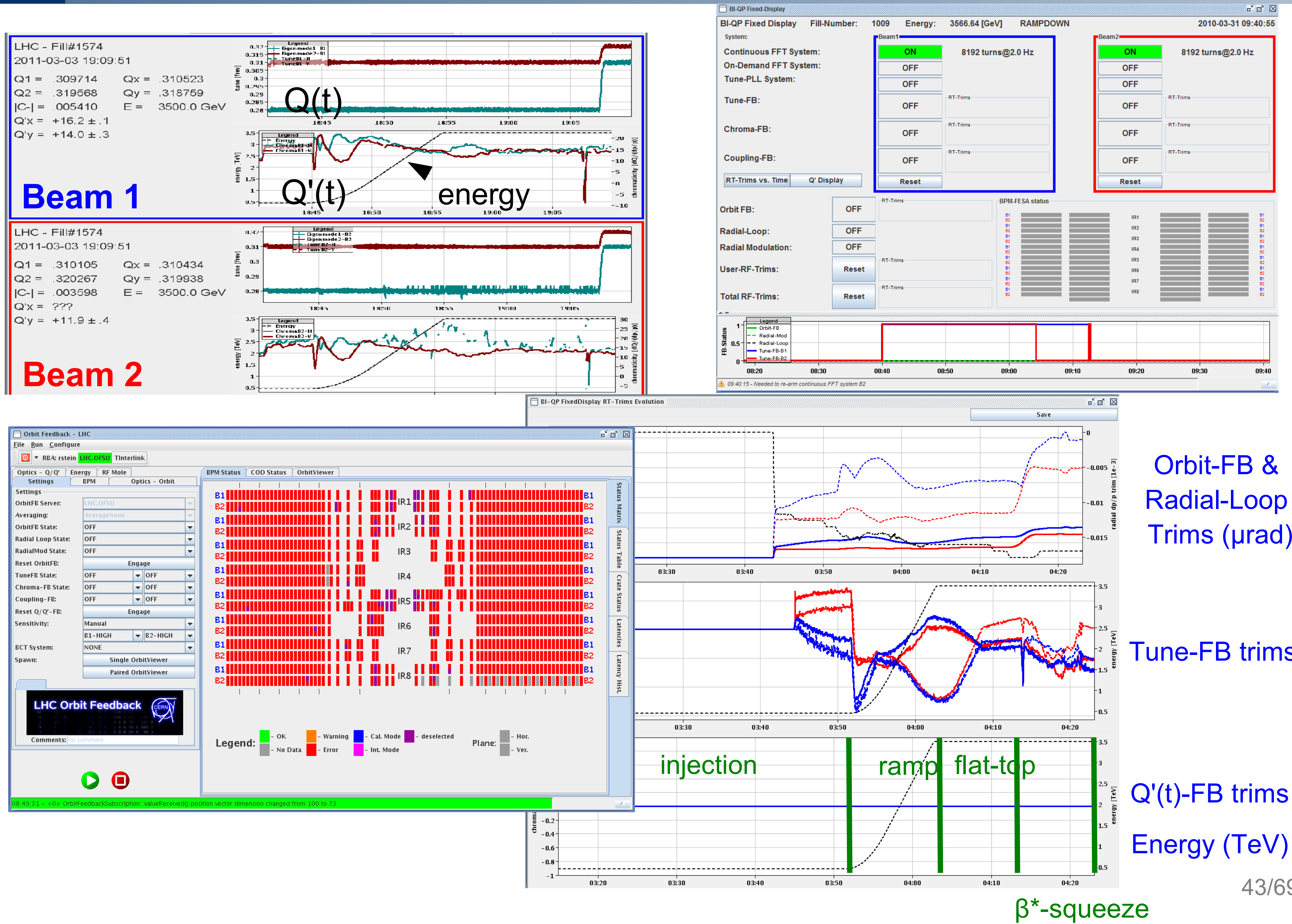
OFC Release Policy

- Since the OFC acts on and directly impact machine operation, any update must be treated as a very sensitive issue (up to MPP-level in some cases)
- Typical steps:
  - Develop, compile, test interfaces against OFSU.DEV
  - Run memory leak, and threading sanity checks (Valgrind, Helgrind, ..)
    - Fix problems if any
  - Run the OFC server for at least 1-2 weeks continuously
    - Monitor CPU and memory footprint, if crash or leak → square one
  - 2-4 weeks before TS announce changes to OP (Jörg, Laurette) and MC!
  - Release version after TS and wait/validate injection sequence and FB response with beam
  - Depending on level of change: test ramp if prescribed by MC/MPP

Feedback Sub-Projects: What they do and where to find them...

- Adopted CO-naming convention, common build style deployment
  - Java well integrated but C++ related part still in progress ...
- In svn+ssh://svn.cern.ch/reps/acc-co/lhc/lhc-feedbacks/
  - lhc-app-orbit-feedback-controller – *the actual feedback controller (aka. OFC)*
  - lhc-lib-feedback-commonalities – glue between various OFC parts and OFSU
    - initially separate feedback controller planned → turned out that this is not possible/recommendable but kept stuff in library to minimise profilling and debugging overhead (rarely changes)
  - lhc-lib-twissoptics – physics/optics related code, not FB dependence per se
  - lhc-lib-twissoptics-examples – examples, documentation and unit-type tests
  - lhc-orbitfeedback – *the OFC/OFSU graphical expert user interface*
  - lhc-app-[orbit/tune]-feedback-serviceunit -- *an orphan FESA class*
  - lhc-orbitfeedback-datamanager -- *reference orbit/sequencer (Kajetan)*
  - lhc-orbitfeedback-services -- *reference orbit/sequencer (Kajetan)*
  - optics-server – *LSA-OFSU link to transfer machine optics data (MAD-X style)*
- two noteworthy exceptions – *Orbit, Q/Q' related GUI (aimed at OP usage)*:
  - svn+ssh://svn.cern.ch/reps/acc-co/lhc/lhc-biqp-fixdisplay/
  - svn+ssh://svn.cern.ch/reps/acc-co/accsoft/tuneviewer

BI-QP Fixed-Display and Orbit Feedback GUI DEMO



Things to be done during the 2011 → 2012 Shutdown
Orbit Diagnostic and FB

- OFC:
  - systematic Orbit-FB energy drift compensation: couldn't identify the cause but internal FB loop on <D·Δx> should cure it
    - some new parameters to control this would need to be exported
    - Change of 'TResponseMatrix' object to include dispersion at CODs
  - Additional BPMs for Diode-Orbit BPM tests
    - Proposal: 'BPMSW.1L1.B1' (WBTN) → 'BPMSWTST.1L1.B1' (DO)
  - Additional status bits flags for permanent and temporary OP mask
- OFSU:
  - More verbosity on generated and sent optics
    - possibility to retrieve and display individual matrices (+ GUI follow-up)
  - Move optics re-computation check/task from OFC to OFSU
    - presently a hack and impedes OFC operation
    - Code-base ready (ResponseOpticsMagic) but needs to be FESA-fied
  - Logging of OFSU/OFC specific IO messages (+ GUI follow-up)
  - Need to shift some expert parameters to OP accessible property
    - FB bandwidth control (RBAC?)
  - Split combined 'Orbit and RF' reset to 'ResetOrbitFB' and 'ResetRFtrims'
  - Pin-down memory leaks...
  - pre-warning: OP indicated request for variable orbit, tune and Q' reference functions, OFC is prepared but some OFSU follow-up required
    - Suggestions for interface/function definition are most welcome!

Things to be tackled during the 2011 → 2012 Shutdown Q/Q' Diagnostics and FB

- Moving the new beam-mode dependent fitter settings from 'ExpertSettings' to 'QfitterSettings'. N.B. Maybe we can find a way to make something similar (time in cycle rather than beam-mode) for the injectors.
- Parallel tune fitter chains
  - cannot find single setting that is optimal for Q,Q', C^- tracking
  - track not only the highest peak but also the following N peaks
    - amplitude, tune-width, S/N ratio estimates would be helpful
  - Needs GUI-follow up, LSA settings integration (TuneViewer and FD)
- Completion of PLL to Linux migration
- Pre-warning: BBQ bunch-selector integration (probably similar to HT gating)
- other items we probably forgot and someone will get upset if we haven't addressed it.

Reserve Slides