COherent Multibunch Beam-Beam Interaction

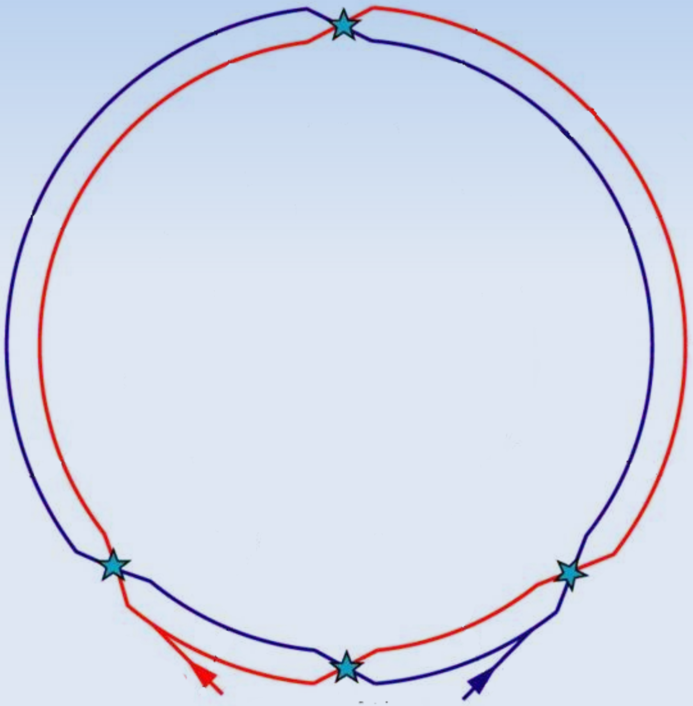
X. Buffat, J. Barranco, T. Pieloni, C. Tambasco

- Physics model
- Code implementation
- Typical usage
- Performance
- Documentation
- License
- Future plans

Physics case

- Two conter rotating beams collide in few interaction points

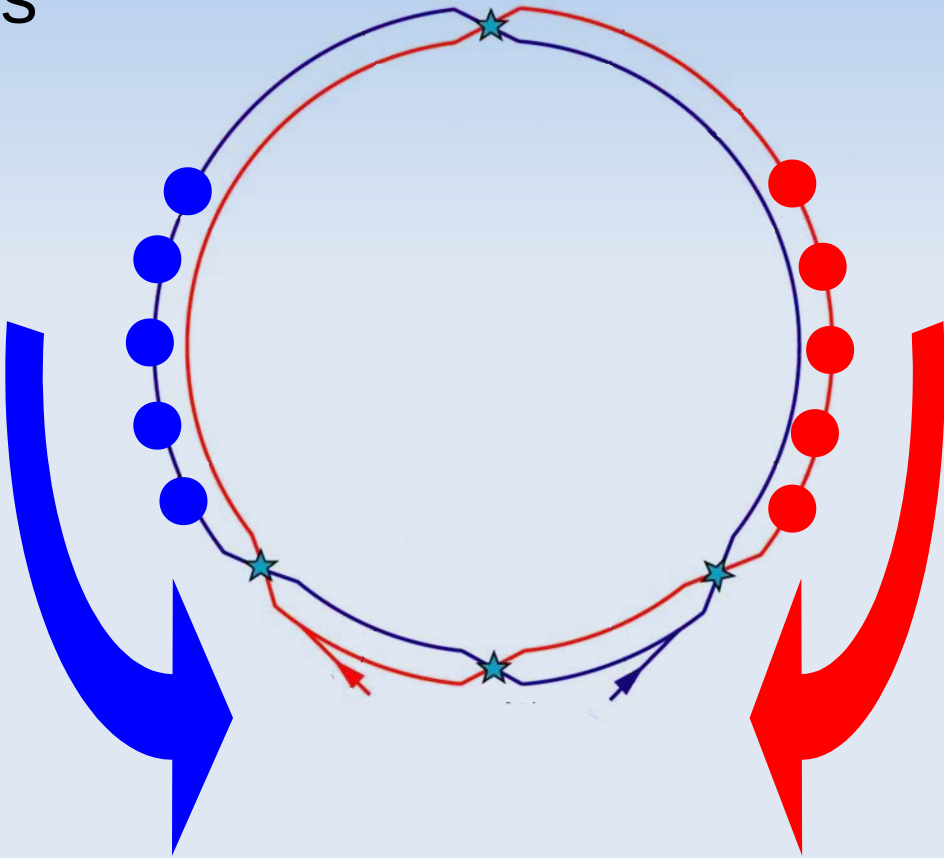
Example : CERN's Large Hadron Collider :



Physics case

- Two conter rotating beams collide in few interaction points
- Each beam is composed of several bunches

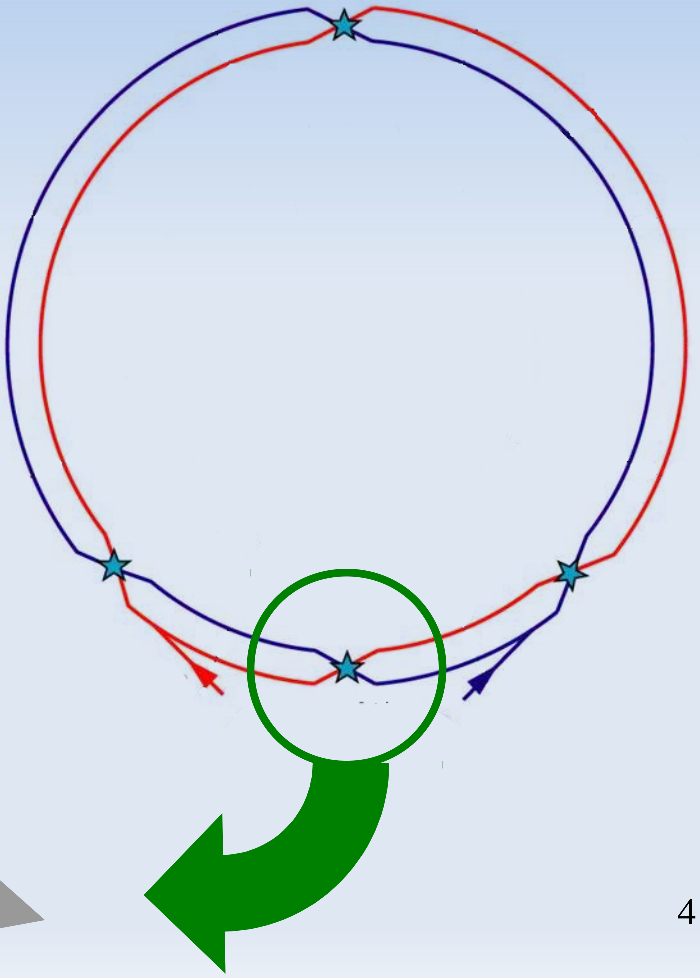
Example : CERN's Large Hadron Collider :

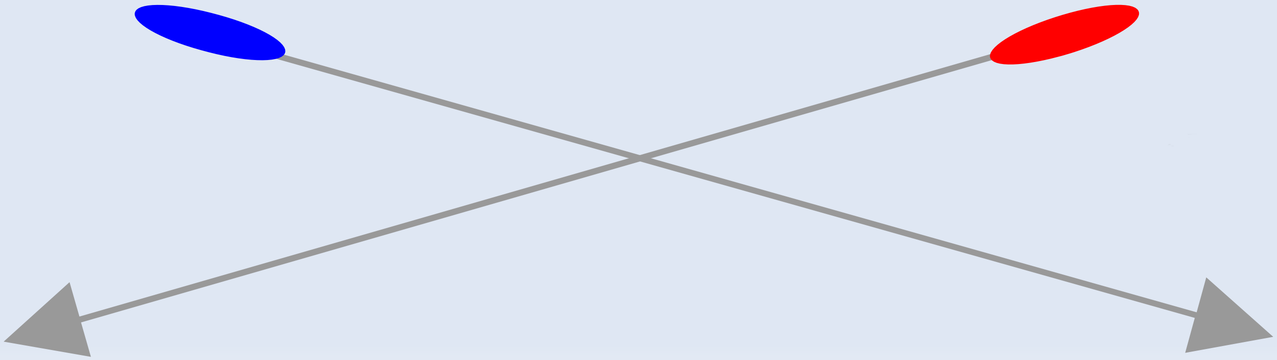


Physics case

- Two conter rotating beams collide in few interaction points
- Each beam is composed of several bunches
- The bunches collide mutliple times each turn

Example : CERN's Large Hadron Collider :

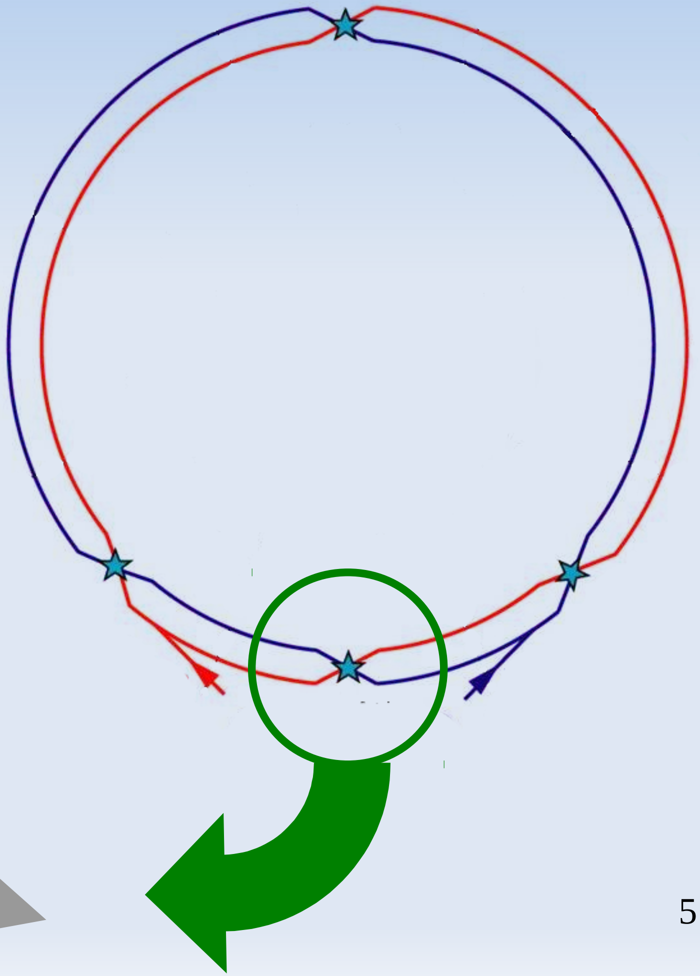


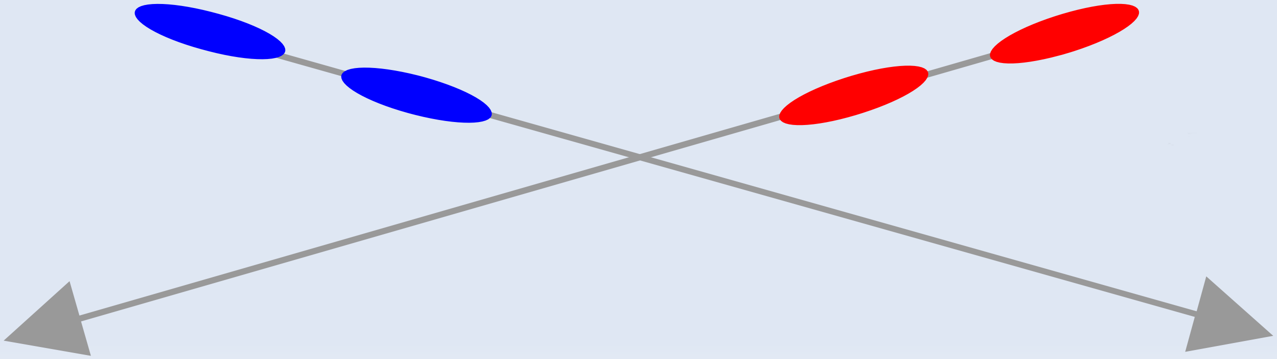


Physics case

- Two conter rotating beams collide in few interaction points
- Each beam is composed of several bunches
- The bunches collide mutliple times each turn

Example : CERN's Large Hadron Collider :

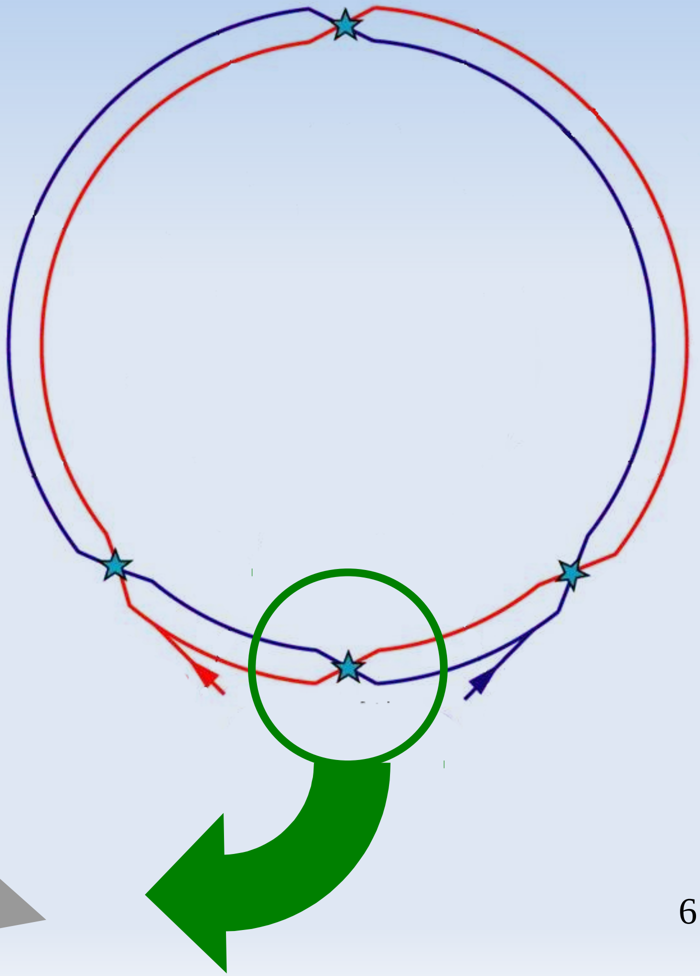


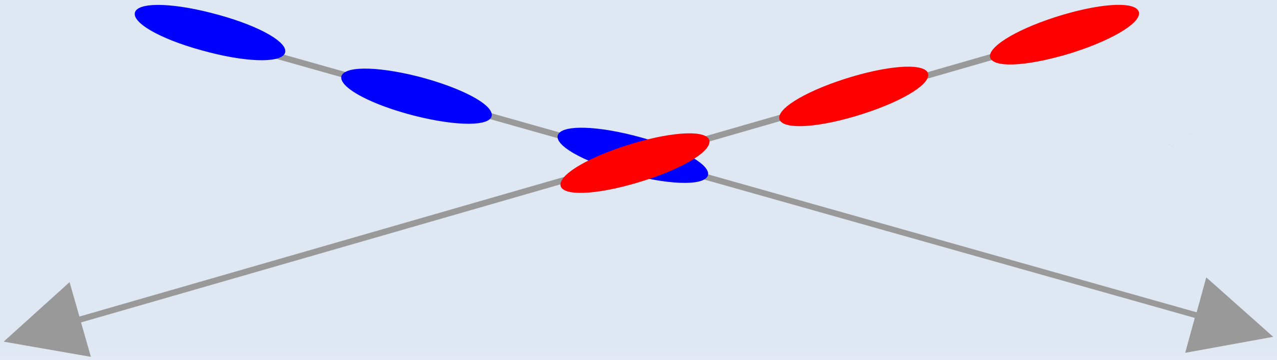


Physics case

- Two conter rotating beams collide in few interaction points
- Each beam is composed of several bunches
- The bunches collide mutliple times each turn

Example : CERN's Large Hadron Collider :

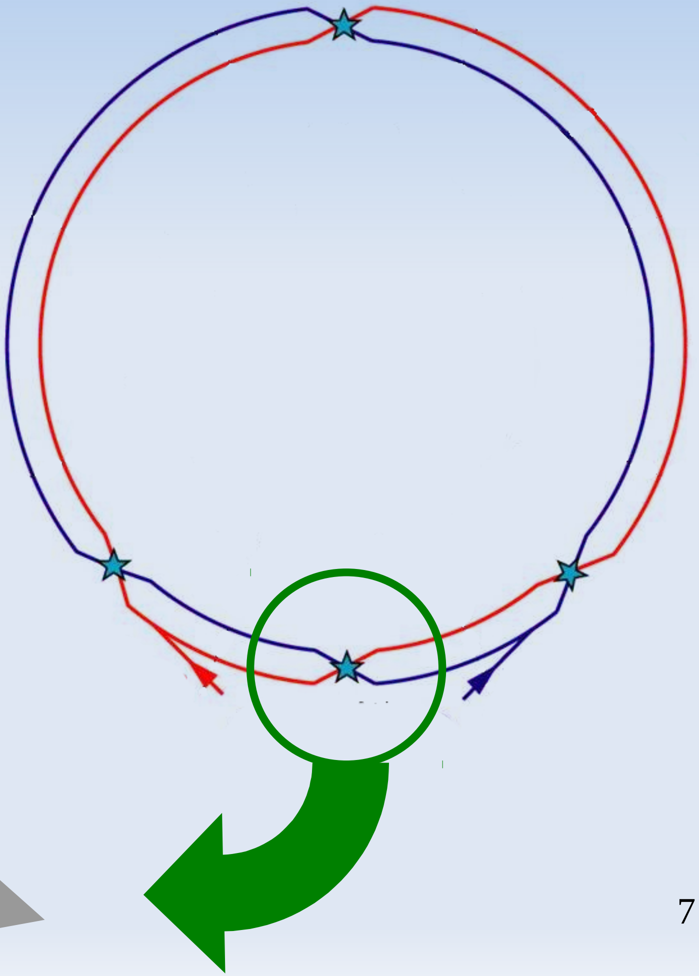


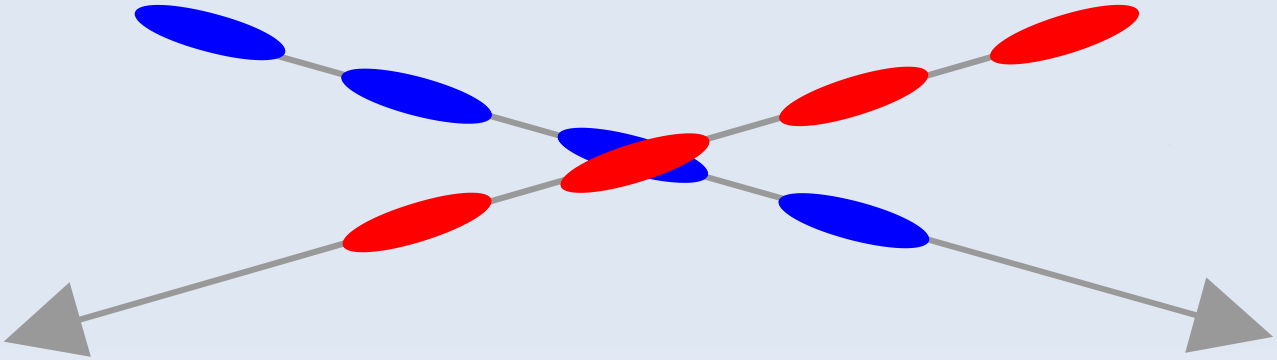


Physics case

- Two conter rotating beams collide in few interaction points
- Each beam is composed of several bunches
- The bunches collide mutliple times each turn

Example : CERN's Large Hadron Collider :

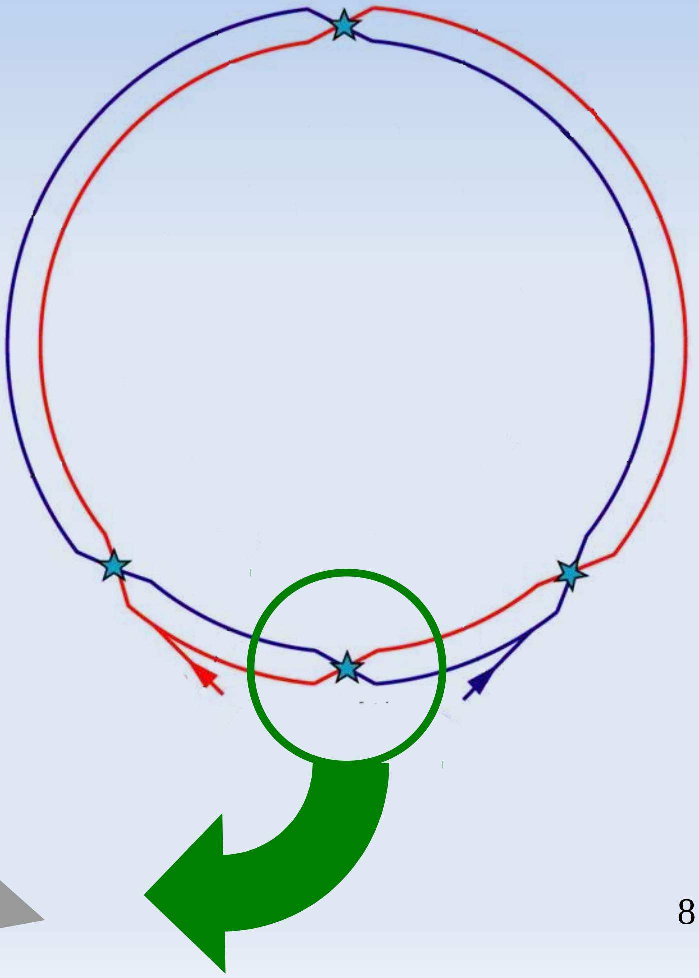


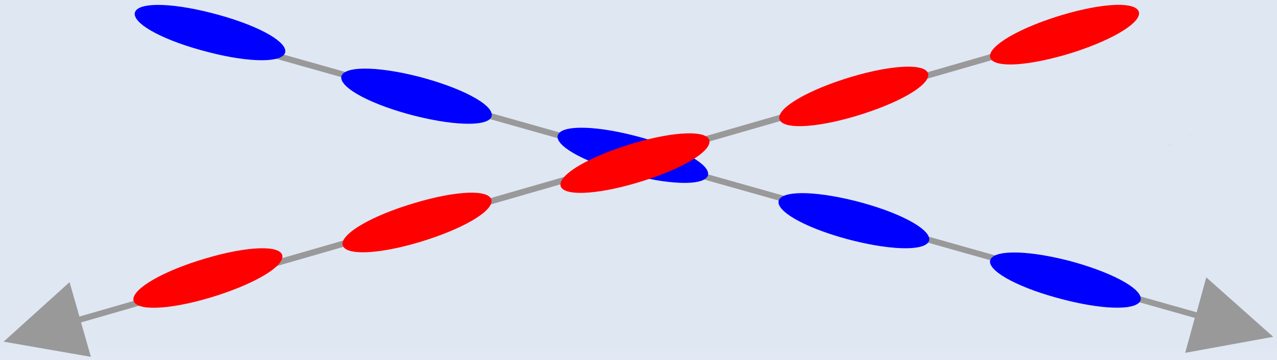


Physics case

- Two conter rotating beams collide in few interaction points
- Each beam is composed of several bunches
- The bunches collide mutliple times each turn

Example : CERN's Large Hadron Collider :

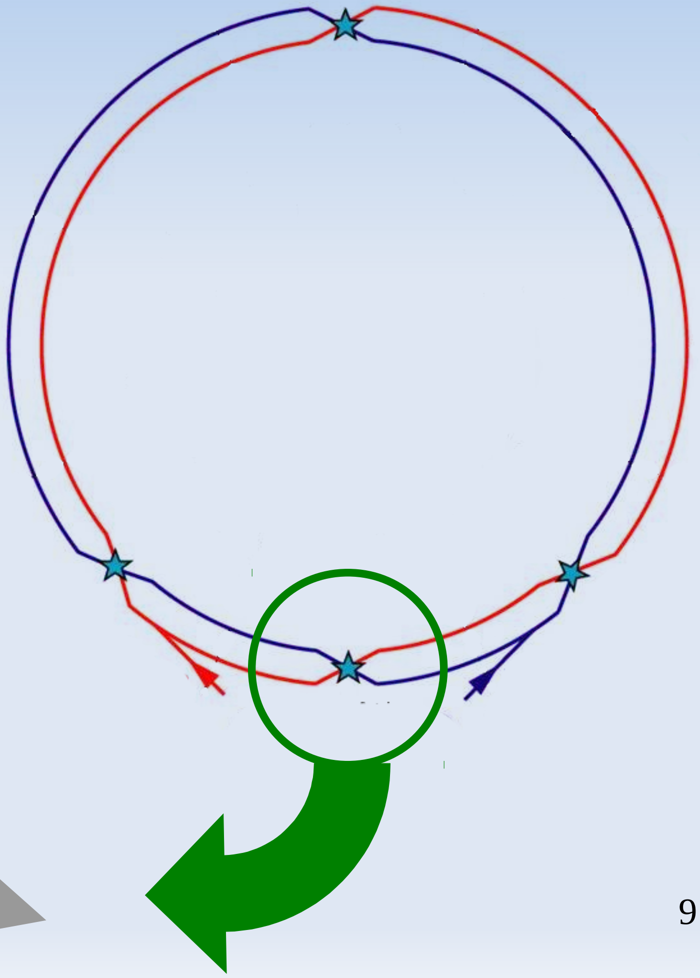


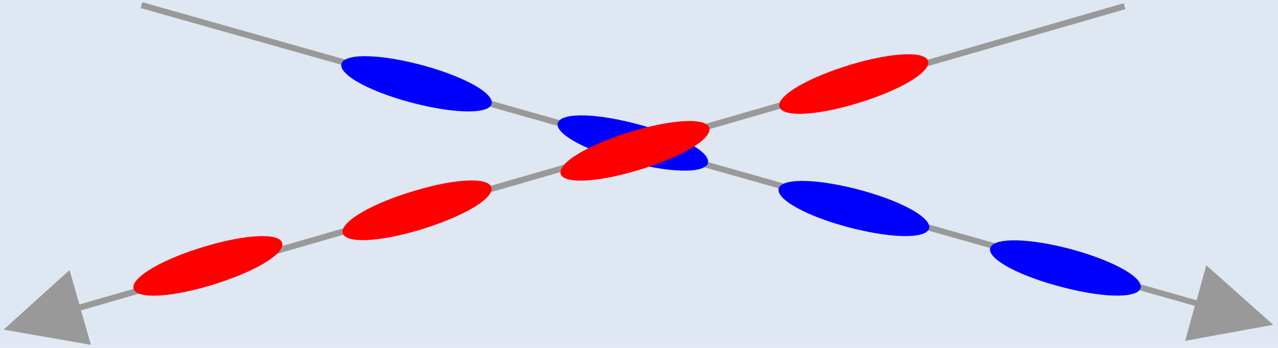


Physics case

- Two conter rotating beams collide in few interaction points
- Each beam is composed of several bunches
- The bunches collide mutliple times each turn

Example : CERN's Large Hadron Collider :

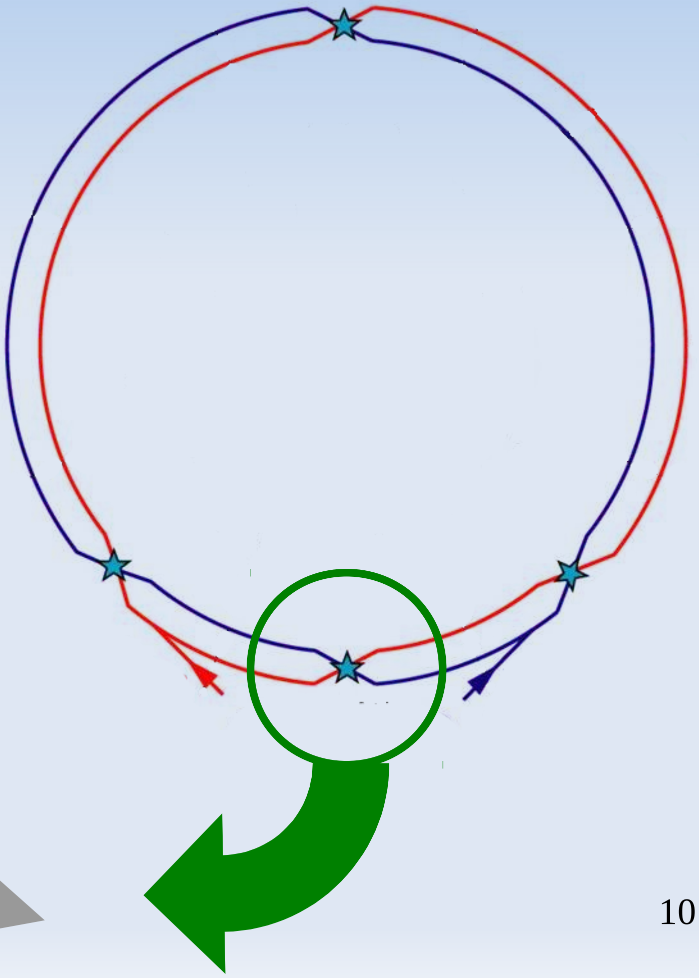


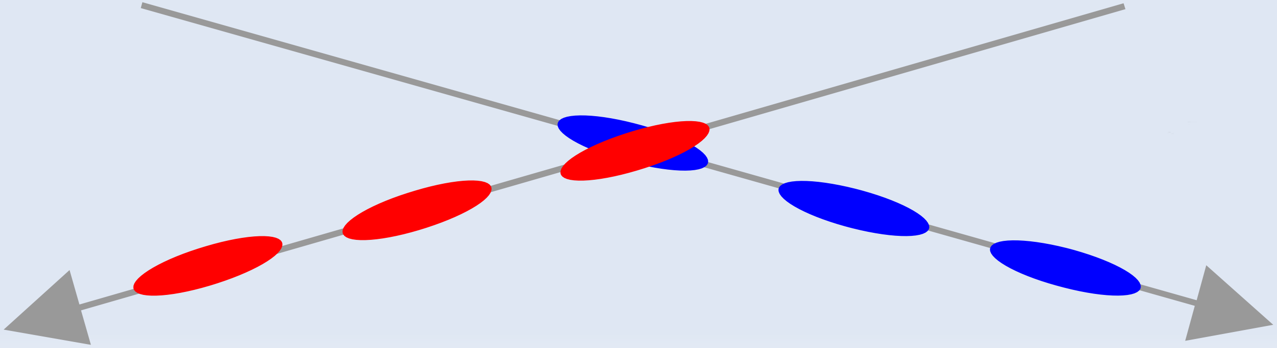


Physics case

- Two conter rotating beams collide in few interaction points
- Each beam is composed of several bunches
- The bunches collide mutliple times each turn

Example : CERN's Large Hadron Collider :

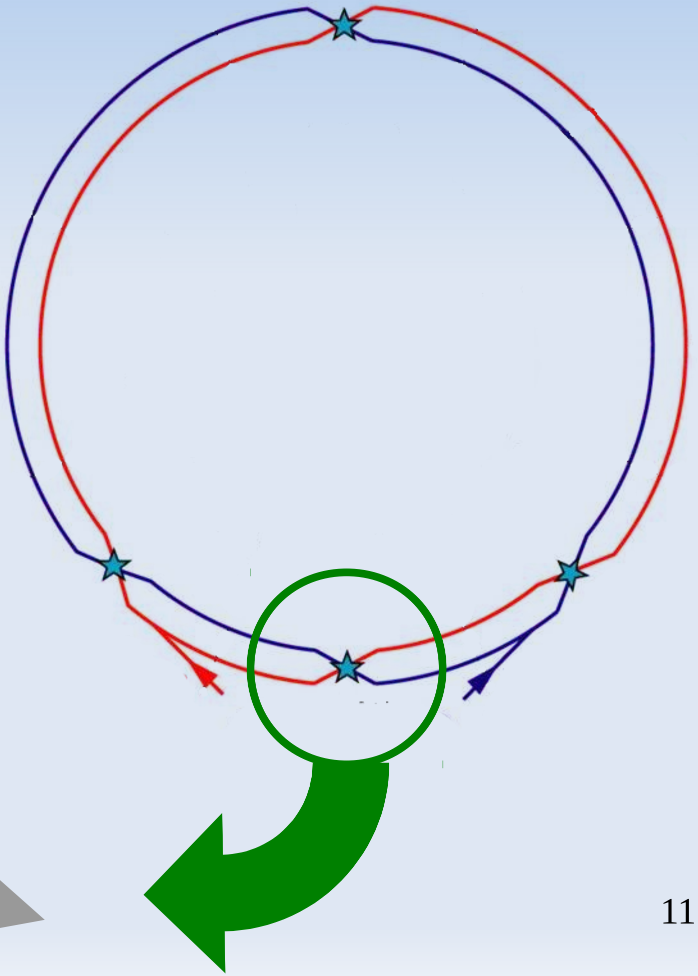


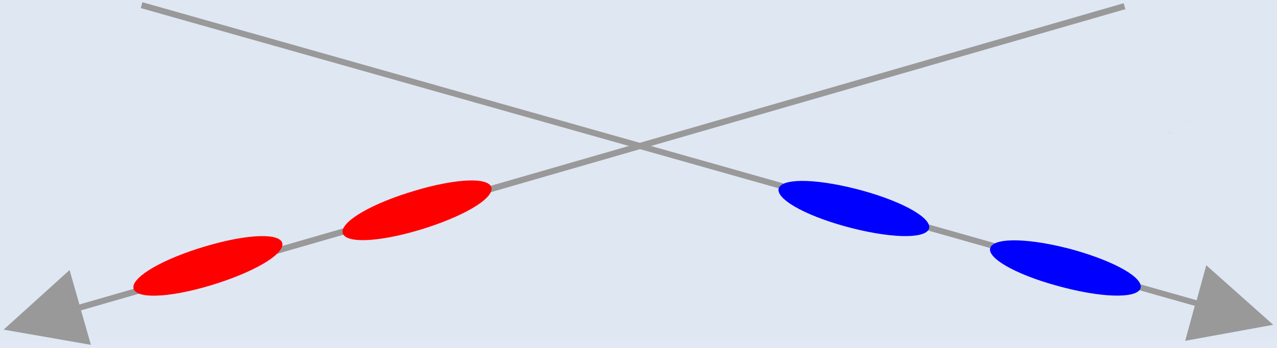


Physics case

- Two conter rotating beams collide in few interaction points
- Each beam is composed of several bunches
- The bunches collide mutliple times each turn

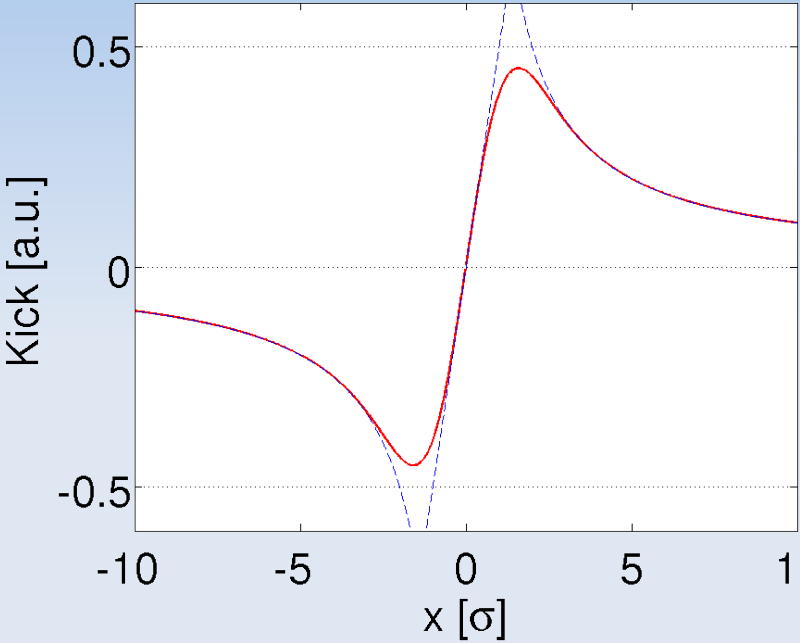
Example : CERN's Large Hadron Collider :

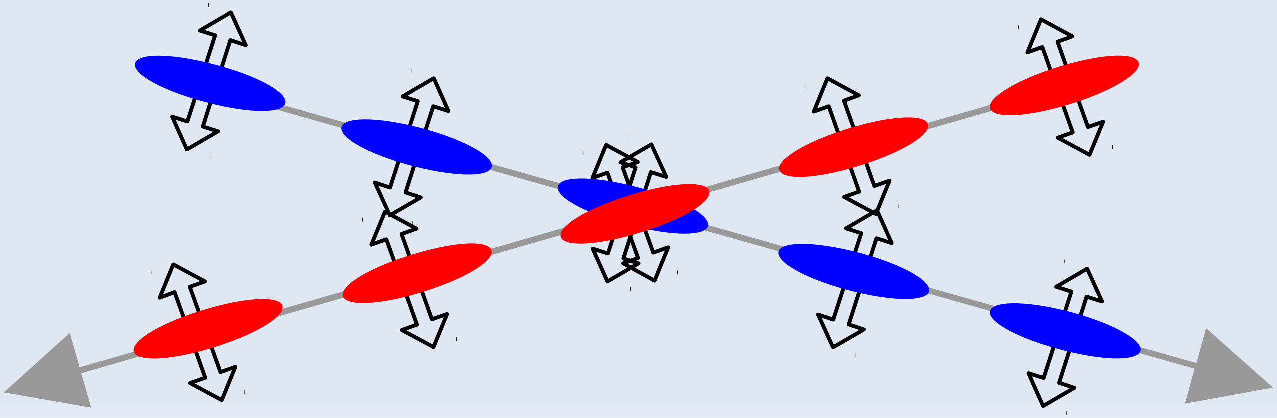




Physics case

- The Beam-beam force
  - depends on the bunch parameters
  - is highly non-linear
  - couples all bunches together
- Many different oscillation mode
- Interplay with other effects



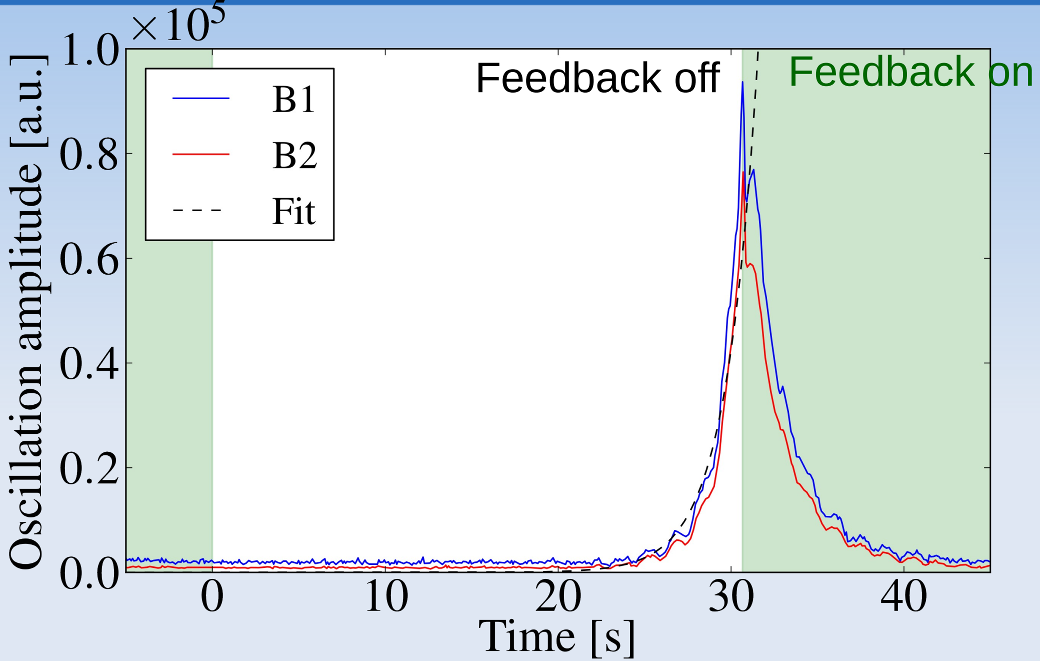


Actions

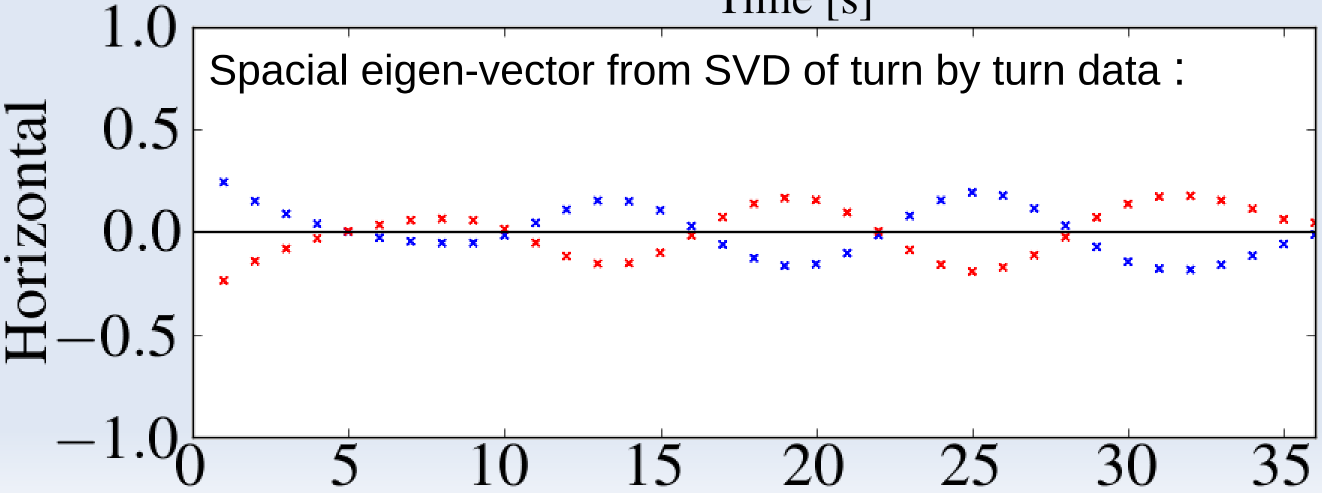
- Linear transport (6D, with chromaticity, linear bucket)
- Head-on collision (HFMM 4D, FPPS 4D, soft-Gaussian 4-6D)
- Long range collision (soft-Gaussian 4D)
- Noise source (white / colored)
- Collimator (”*in or out* ” model)
- Impedance (equidistant slices and wake tables)
- Linear detuning (e.g. due to octupole)
- Transverse feedback (perfect, ADT-like)
- Synchrotron radiation (damping, quantum excitation)
- ...

Impact on CERN machines
Colliding beam stability

- Observation of coupling instability of colliding beams in the LHC, damped by the transverse feedback in agreement with COMBI simulations



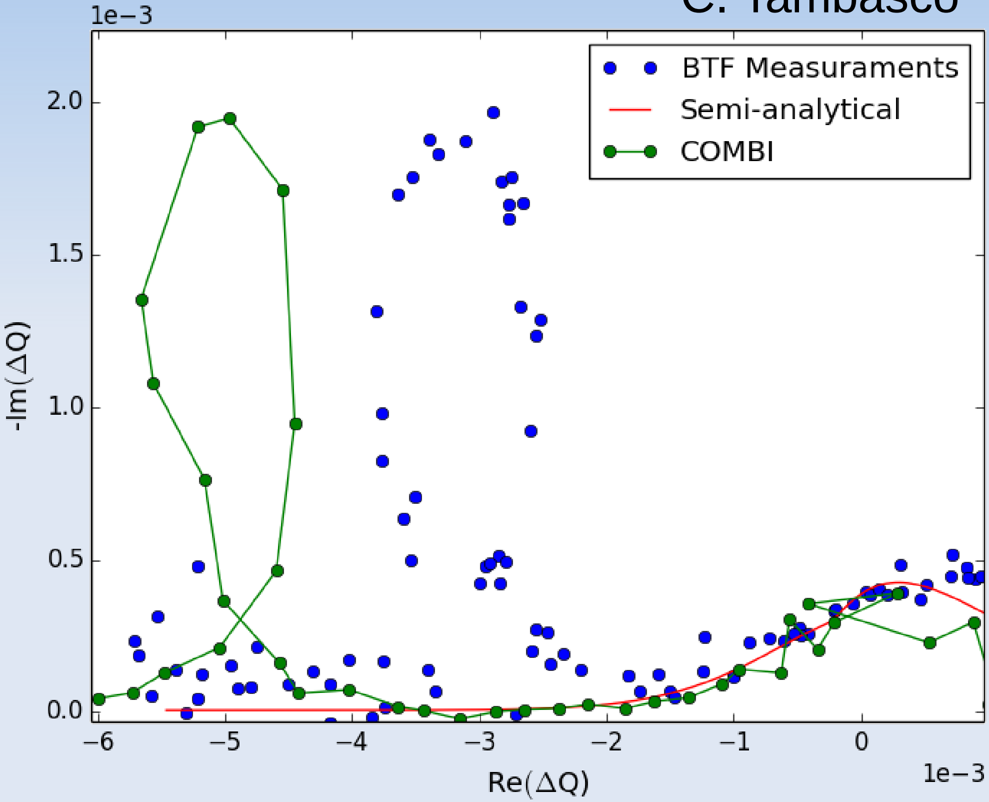
- Simulation of coupled bunch - coupled beam instability of two trains of 36 bunches colliding long-range



Impact on CERN machines
Beam transfer function

- Beam transfer function measurements are used as a diagnostic tool to quantify Landau damping
- Can simulate complex features that are not accessible with the analytical models (e.g. chromaticity, distorted particle distributions)

C. Tambasco

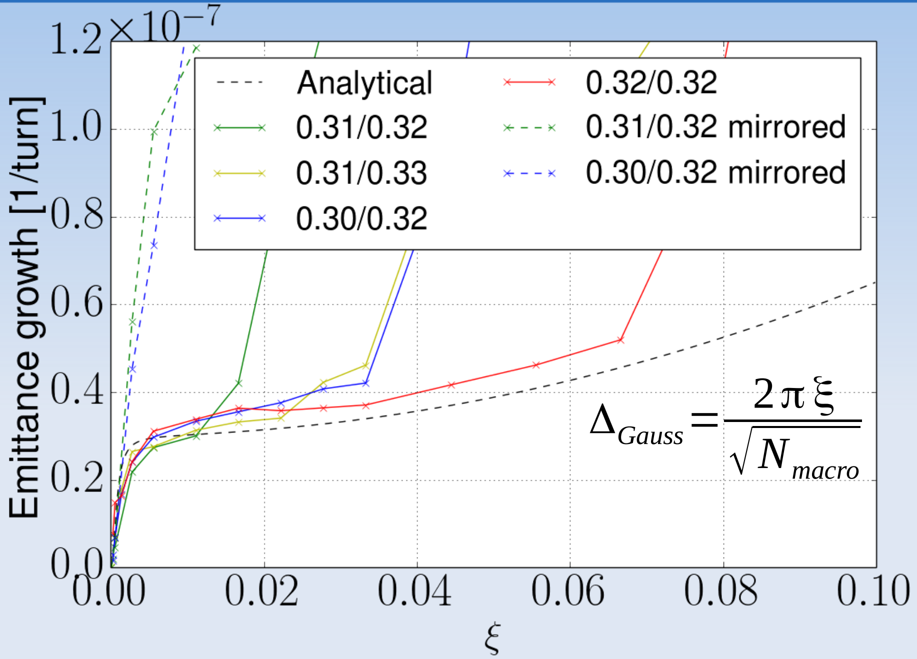


Impact on CERN machines
Emittance effects

- Noise on colliding beams generates an important emittance growth

  → specification for HL-LHC crab cavities

  → specification for PC ripple, ground motion tolerance, etc (HL-LHC, FCC-hh)



$$\Delta_{Gauss} = \frac{2\pi\xi}{\sqrt{N_{macro}}}$$

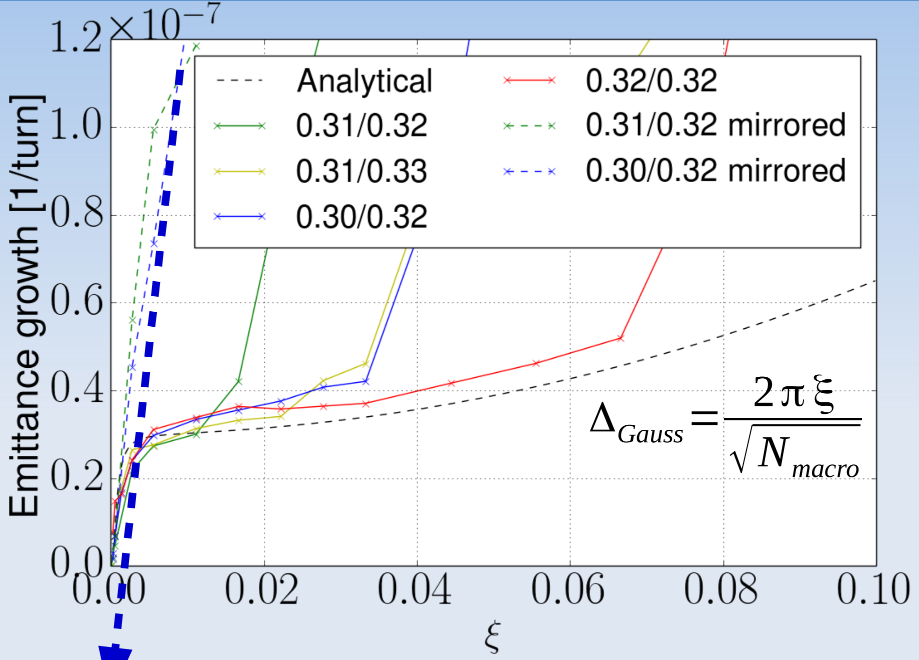
- Can simulate complex features that are not accessible with the analytical models (e.g. 2nd order effects, tune effects, chromaticity, …)
- Identify other observables to test the models
- Solver noise is critical for this type of study

Impact on CERN machines
Emittance effects

- Noise on colliding beams generates an important emittance growth

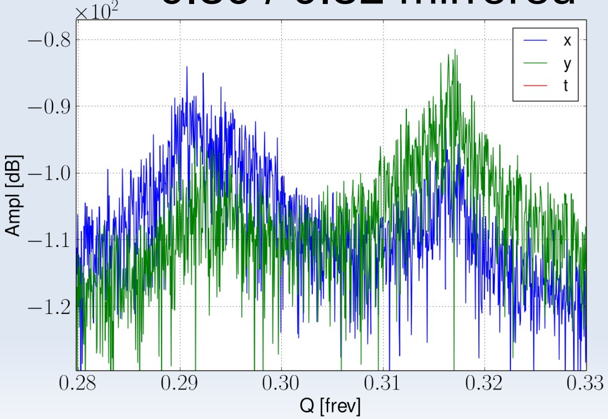
  → specification for HL-LHC crab cavities

  → specification for PC ripple, ground motion tolerance, etc (HL-LHC, FCC-hh)



- Can simulate complex features that are not accessible with the analytical models (e.g. 2nd order effects, tune effects, chromaticity, …)
- Identify other observables to test the models
- Solver noise is critical for this type of study

0.30 / 0.32 mirrored

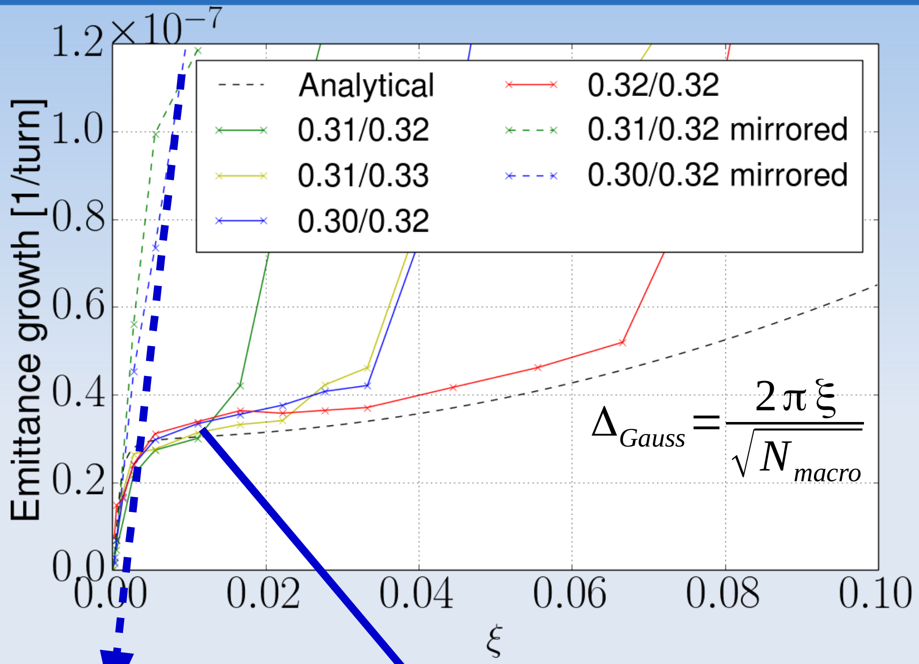


Impact on CERN machines
Emittance effects

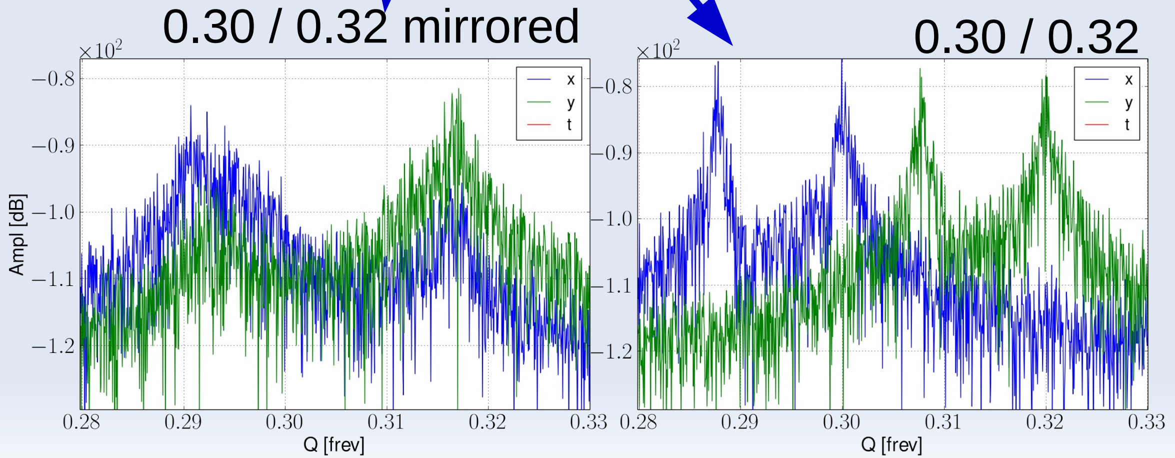
- Noise on colliding beams generates an important emittance growth

  → specification for HL-LHC crab cavities

  → specification for PC ripple, ground motion tolerance, etc (HL-LHC, FCC-hh)



- Can simulate complex features that are not accessible with the analytical models (e.g. 2nd order effects, tune effects, chromaticity, …)
- Identify other observables to test the models
- Solver noise is critical for this type of study


0.30 / 0.32 mirrored

0.30 / 0.32

Head-on and long range beam-beam interactions
Soft-gaussian approximation (4D)

- Need to solve Poisson equation in 2D (open boundary) twice per interaction per turn (i.e. efficiently): several methods exist

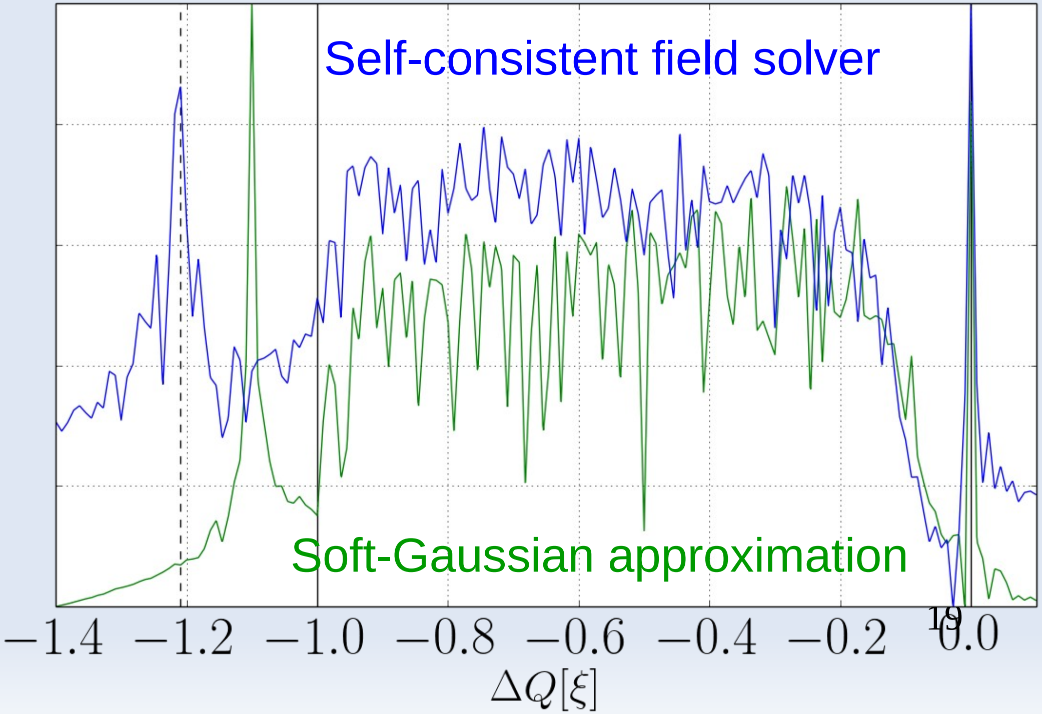
- Compute first order moments of the distribution and compute the field based on a Gaussian distribution :

$$\Delta x' = \frac{-2\mathrm{r}_0 N}{\gamma_r} \frac{x}{r^2}\left(1 - e^{\frac{-r^2}{2\sigma^2}}\right)$$

- Fastest and most precise solver, yet it is not accurate especially for non-Gaussian beams

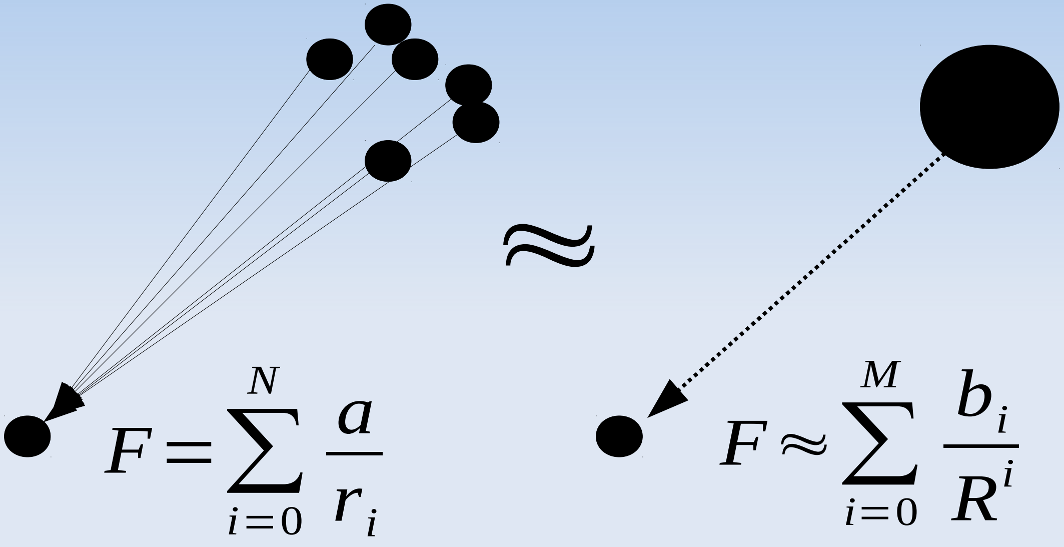
- Used to compute long-range beam-beam interactions since the dependency on the details of the beam distribution is less critical

- Sufficient for most coherent stability studies

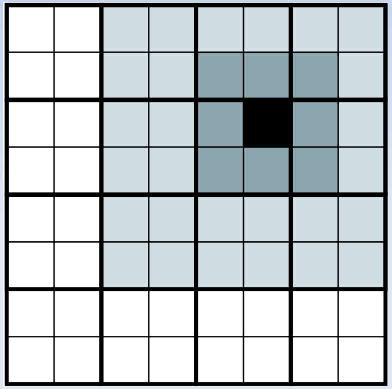


Head-on beam-beam interaction
Fast Multipole Method (4D)

- Group far particles and approximate the kick with a multipolar expansion :

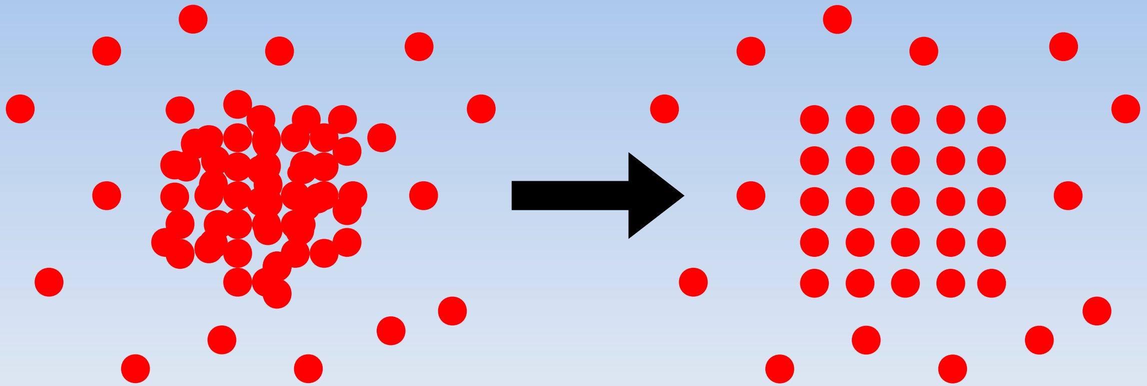


$$F=\sum_{i=0}^{N}\frac{a}{r_i} \approx \quad F\approx\sum_{i=0}^{M}\frac{b_i}{R^i}$$



- Smart bookkeeping allows to do reduce the complexity from $O(N^2)$ to O(N)
- Accurate (≠precise) for any type of distribution
- Still quite slow for large particle density, especially when the beams are slightly separated
- In a parallel implementation, the position and charge of all particles have to be exchanged between processes, could be slow in clusters without shared memory

Head-on beam-beam interaction
Hybrid FMM (4D)



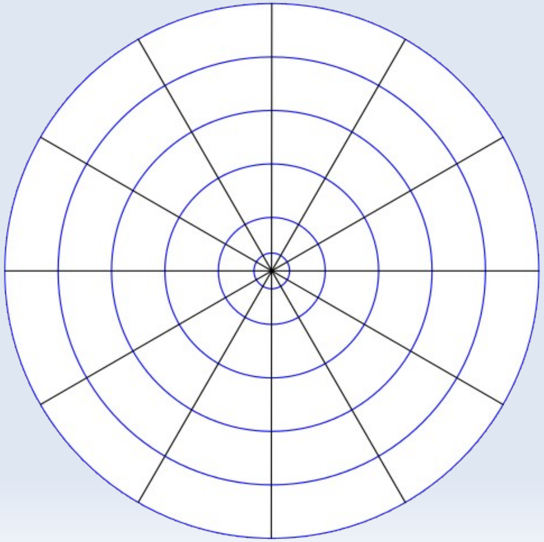
- Distribute the charge of the core on a rectangular mesh, to reduce the computing time and the amount of data to be transfered
  - Grid properties are set with preprocessor directives (need to recompile!)
- Initially designed for space charge, the HFMM was also used in BEAMX (single 6D beam-beam interaction)
- Working horse of COMBI, since it was heavily benchmarked against analytical formulas, other codes (BeamBeam3D) and observations at RHIC and LHC

HFMM maintainability

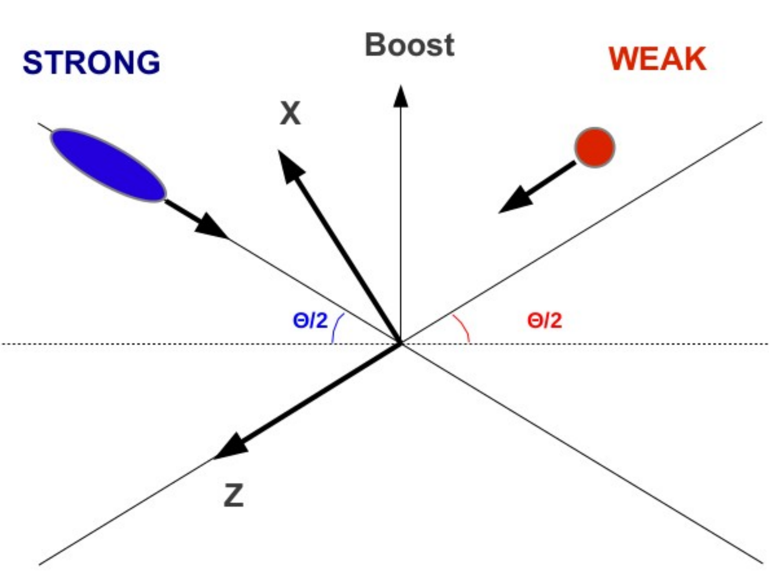
- The implementation of the HFMM is robust and fast for most application, but is difficult to maintain and would require major refactoring to be parallelisable (and probably wouldn't be efficient anymore!)
- The quad tree algorithm is very efficient, but really noisy
  - Shows a saturation of the noise level at large number of macroparticle (>10^6) that is not compatible with present needs
- We (A. Florio) looked into possibilities to improve :
  - Parallel version of the FMM, with a fixed grid (F2M2)
  - Fast Polar Poisson Solver (FPPS)

Head-on beam-beam interaction
Fast Polar Poisson Solver (4D)

- Distribute the charge on a polar mesh, solve Poisson on the angular coordinate using the FFT and finite differences on the radial coordinate
  - Removes the need for copies as well as the artifacts introduced with a FFT solver on a 2D Cartesian mesh
  - Stretch the radial coordinate to simulate open-boundary condition
  - Fast, accurate and more precise than the HFMM
  - Trivial implementation of the second level of parallelisation with OpenMP
  - Singularity at the center is not trivial to handle



Head-on beam-beam interaction
Soft-Gaussian 6D model

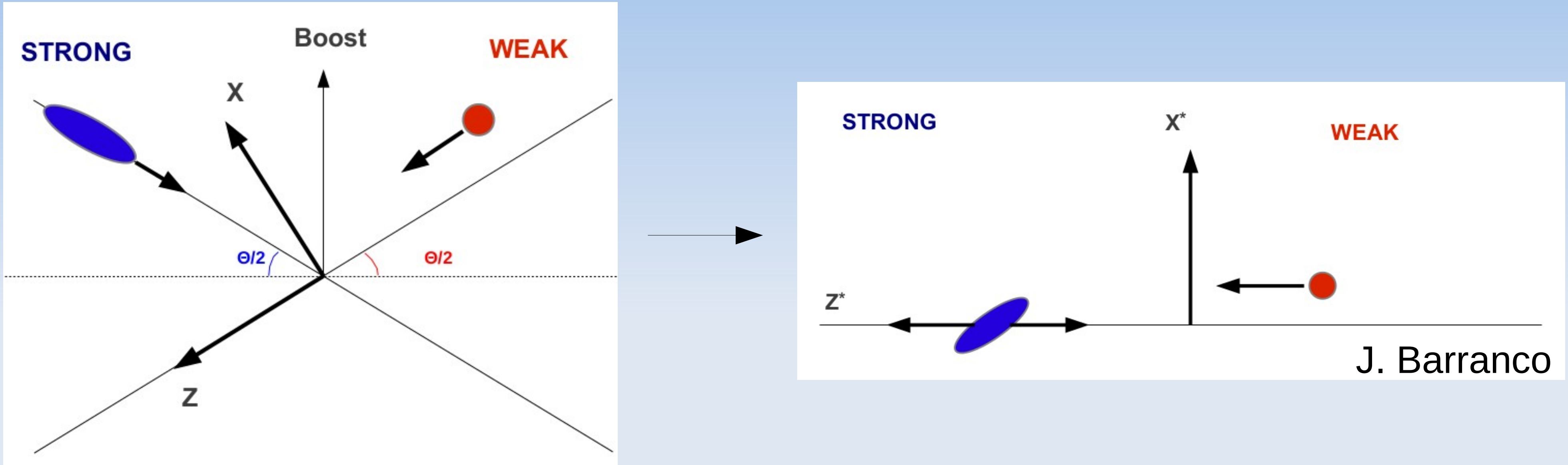


- Based on K. Hirata's BBC

  → Compute the weak-strong beam-beam kick for all particles, taking the other beam's longitudinal slices' first order moments to model the strong beam (soft-Gaussian)

- Note : A fully self-consistent 6D solver is implemented in BEAMX (HFMM) and BeamBeam3D (FFT) → higher computing requirements

Head-on beam-beam interaction
Soft-Gaussian 6D model



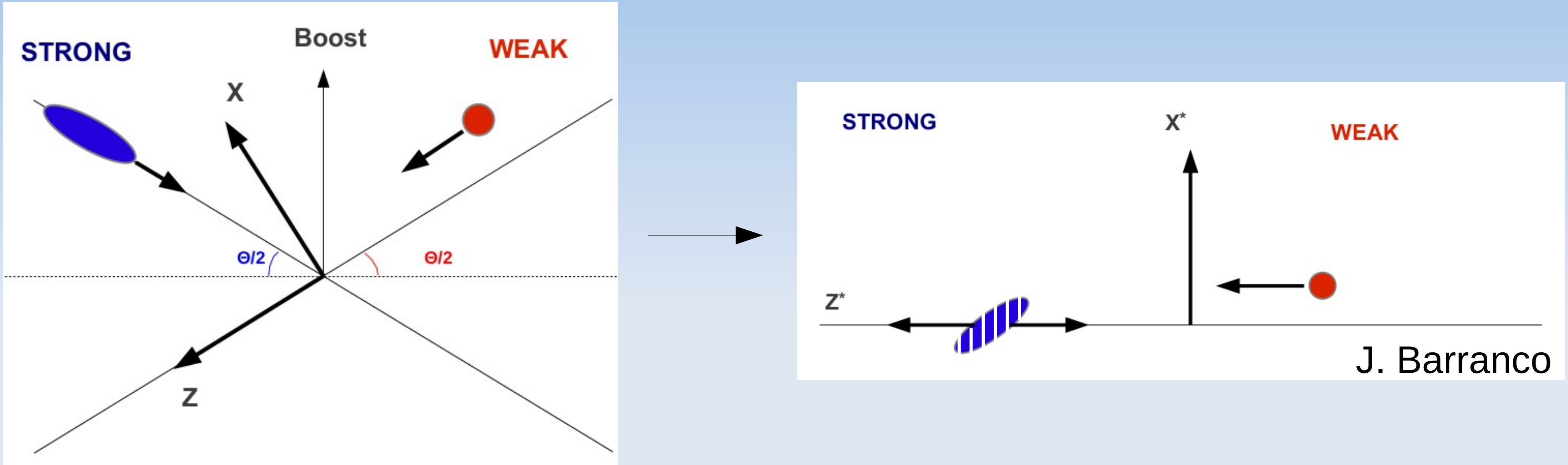
J. Barranco

- Based on K. Hirata's BBC

→ Compute the weak-strong beam-beam kick for all particles, taking the other beam's longitudinal slices' first order moments to model the strong beam (soft-Gaussian)

- Note : A fully self-consistent 6D solver is implemented in BEAMX (HFMM) and BeamBeam3D (FFT) → higher computing requirements

Head-on beam-beam interaction
Soft-Gaussian 6D model



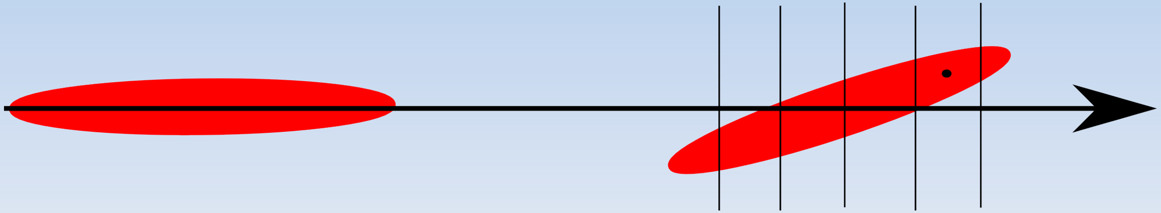
J. Barranco

- Based on K. Hirata's BBC

→ Compute the weak-strong beam-beam kick for all particles, taking the other beam's longitudinal slices' first order moments to model the strong beam (soft-Gaussian)

- Note : A fully self-consistent 6D solver is implemented in BEAMX (HFMM) and BeamBeam3D (FFT) → higher computing requirements

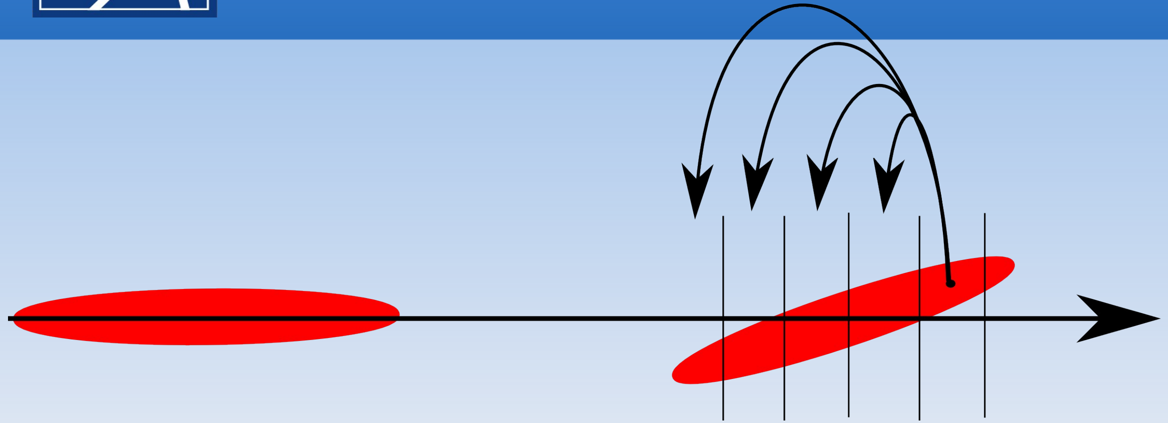
Impedance

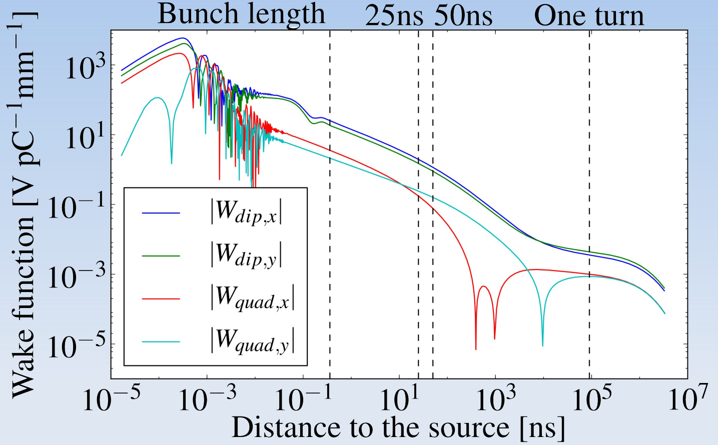


- A la HEADTAIL :
  - Slice the bunch longitudinally (equidistant)
  - Compute the charge and average positions of each slice
  - Apply the kicks to trailing particles and trailing bunches based on the wake function (wake table or resonator implemented in the development branch)

    → Requires *long term* (i.e. few turns) data storage and communication between bunches → see later
  - Only single kick per turn possible
  - Benchmarked with HEADTAIL multibunch (N. Mounet) (TMCI, coupled bunch instability rise times, octupole thresholds)

Impedance

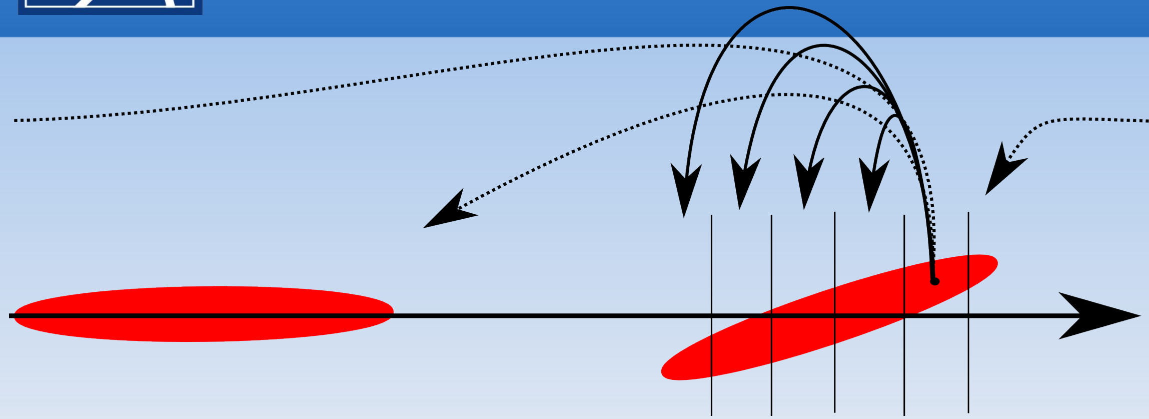


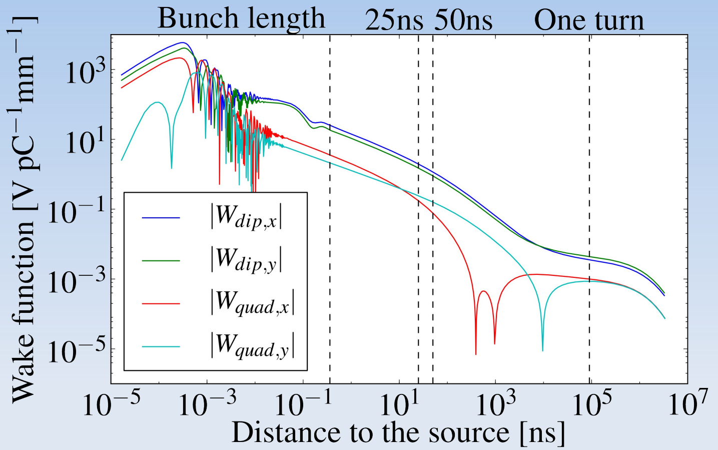


- A la HEADTAIL :
  - Slice the bunch longitudinally (equidistant)
  - Compute the charge and average positions of each slice
  - Apply the kicks to trailing particles and trailing bunches based on the wake function (wake table or resonator implemented in the development branch)

    → Requires *long term* (i.e. few turns) data storage and communication between bunches → see later
  - Only single kick per turn possible
  - Benchmarked with HEADTAIL multibunch (N. Mounet) (TMCI, coupled bunch instability rise times, octupole thresholds)

Impedance





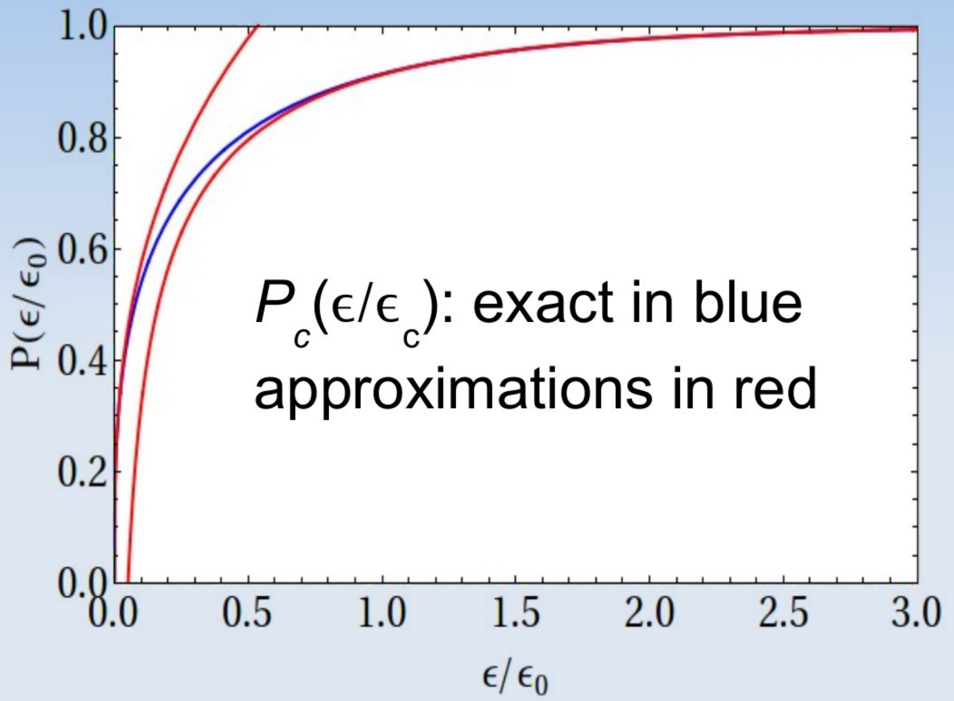
- A la HEADTAIL :
  - Slice the bunch longitudinally (equidistant)
  - Compute the charge and average positions of each slice
  - Apply the kicks to trailing particles and trailing bunches based on the wake function (wake table or resonator implemented in the development branch)

    → Requires *long term* (i.e. few turns) data storage and communication between bunches → see later
  - Only single kick per turn possible
  - Benchmarked with HEADTAIL multibunch (N. Mounet) (TMCI, coupled bunch instability rise times, octupole thresholds)

Synchrotron radiation

- Full implementation (Based on S. White's):
  - For each particle at each arc (or each turn), compute the number of photons emitted based on Poisson distribution
  - Compute each photon energy based on probability density (analytical + lookup table)
  - Apply longitudinal and transverse kicks
- Gaussian noise model (default)
  - When several photons are emitted per arc (or per turn), the energy loss per particle becomes Gaussian distributed

    → Apply averaged damping on all particles, with a single particle Gaussian white noise (longitudinal and transverse) (based on either the radiation integrals or the equilibrium emittance and damping time)



$P_c(\epsilon/\epsilon_c)$: exact in blue approximations in red

Code implementation

- Some old (1987) routines are written in F77 (HFMM, BBC), the newest features are implemented as C++ class (FPPS*)
  - Most features are implemented in F90 or C
- The wrapping and the first level of parallelisation (MPI) (→ combi.c, master.c, slave.c) is written in C
- Compiled and tested with gcc and icc only
- A second level of parallelisation based on OpenMP is implemented in all functions (only where it offers a gain)
  - Requires an implementation of MPI with the level of thread support MPI_THREAD_FUNNELED (note: usually MPI_THREAD_SINGLE still works, but no guarantees)
- The FPPS relies on FFTW3

* Pythonized version available in PyPIC

Master and slave initialisation

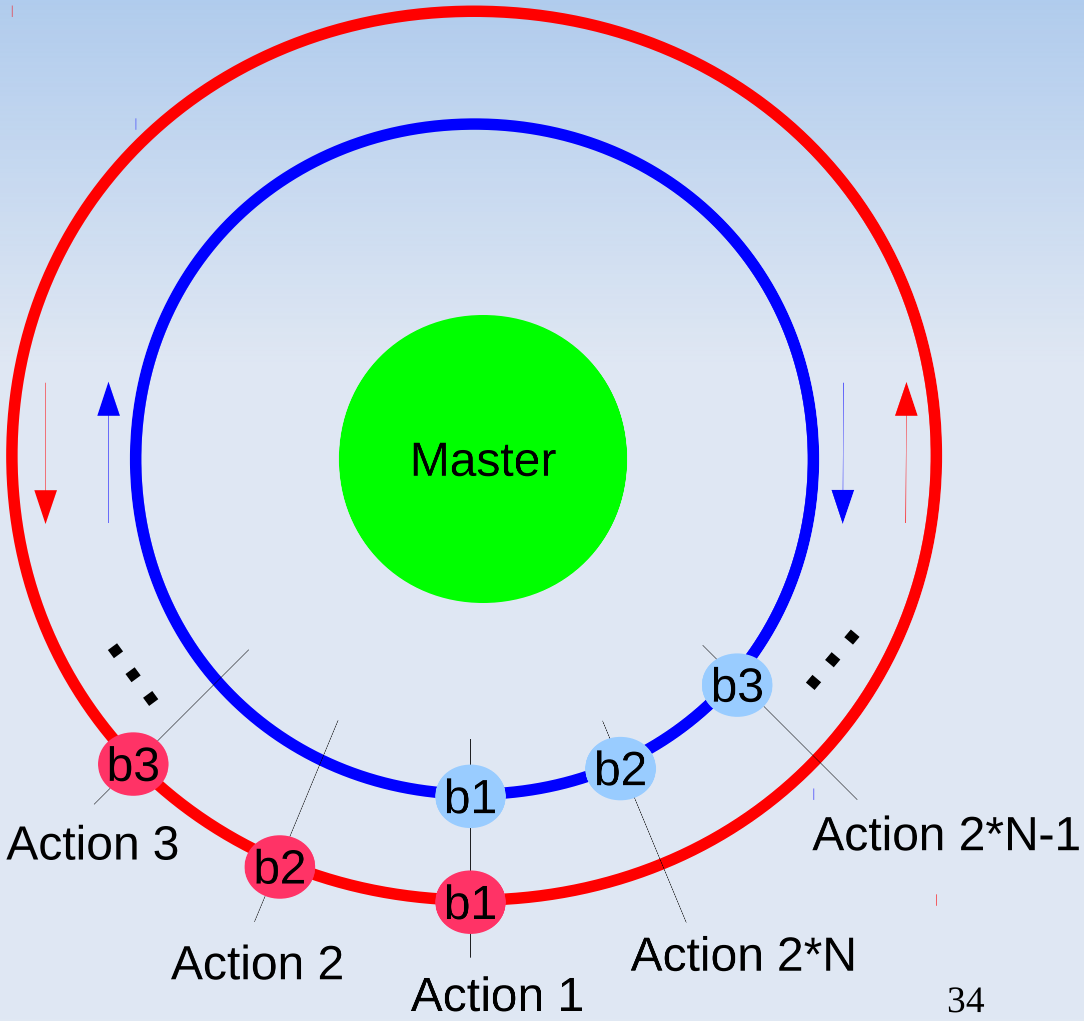
- At initialisation the process with ID 0 becomes the masters (combi.c → master.c)
  - Reads input files (checks their integrity)
  - Makes a mapping MPI ID to bunches, cancels unused processes
  - Sends initialisation data to slaves and waits for return values of all slaves
  - Do one test turn (check integrity, compute collision pattern, both the slave and the master allocate the memory that they will need during the execution)
- Other process are slaves (combi.c → slave.c)
  - Read input files
  - Allocate memory for the beam (array of double representing 7 coordinates (6D phase space + charge) of each particle
  - Enter the 'while(1)' loop :
    - Wait for an instruction from the master (action code, …)
    - Execute (slave.c → Fortran / C / C++ functions)
    - Send completion message

Input

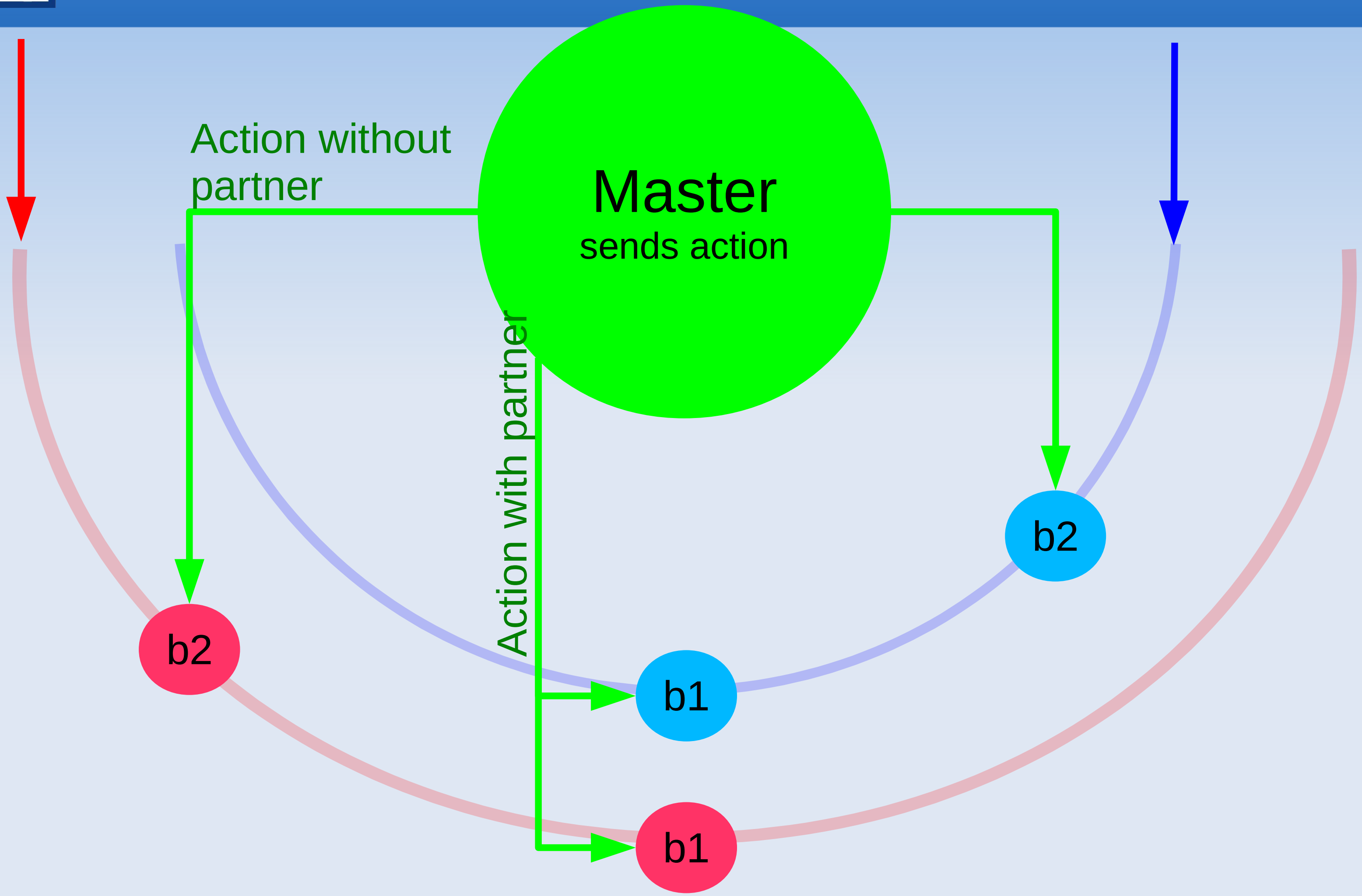
- *.in
  - Name of other input files
  - Select type of output and set output file names
  - Machine and beam parameters
- *.fill
  - Define the bunch configuration (i.e. filling scheme in the LHC)
- *.coll
  - Define the list of actions representing the machine (equivalent to the sequence in MAD, but for two interacting beams)

First level of parallelization

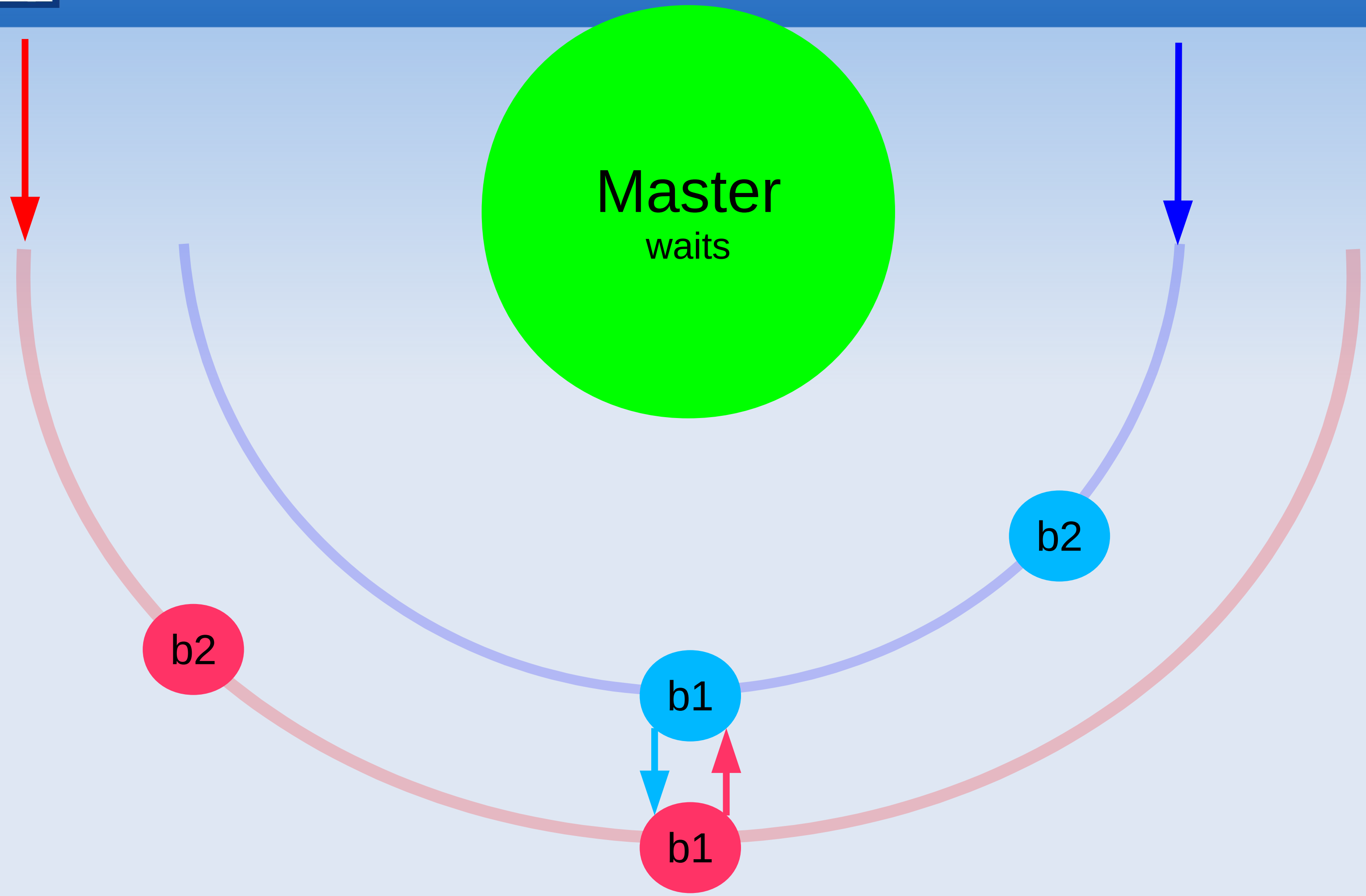
- Usually, the number of bunches is larger than available CPU per node → MPI required
  - 1 **master** process
  - 1 **slave** process per bunch
- Both beams go through the same sequence of action, but in opposite direction
  - It is the responsibility of the user (with the help of few helper output at initialisation) to ensure the consistency between the filling scheme and the action sequence



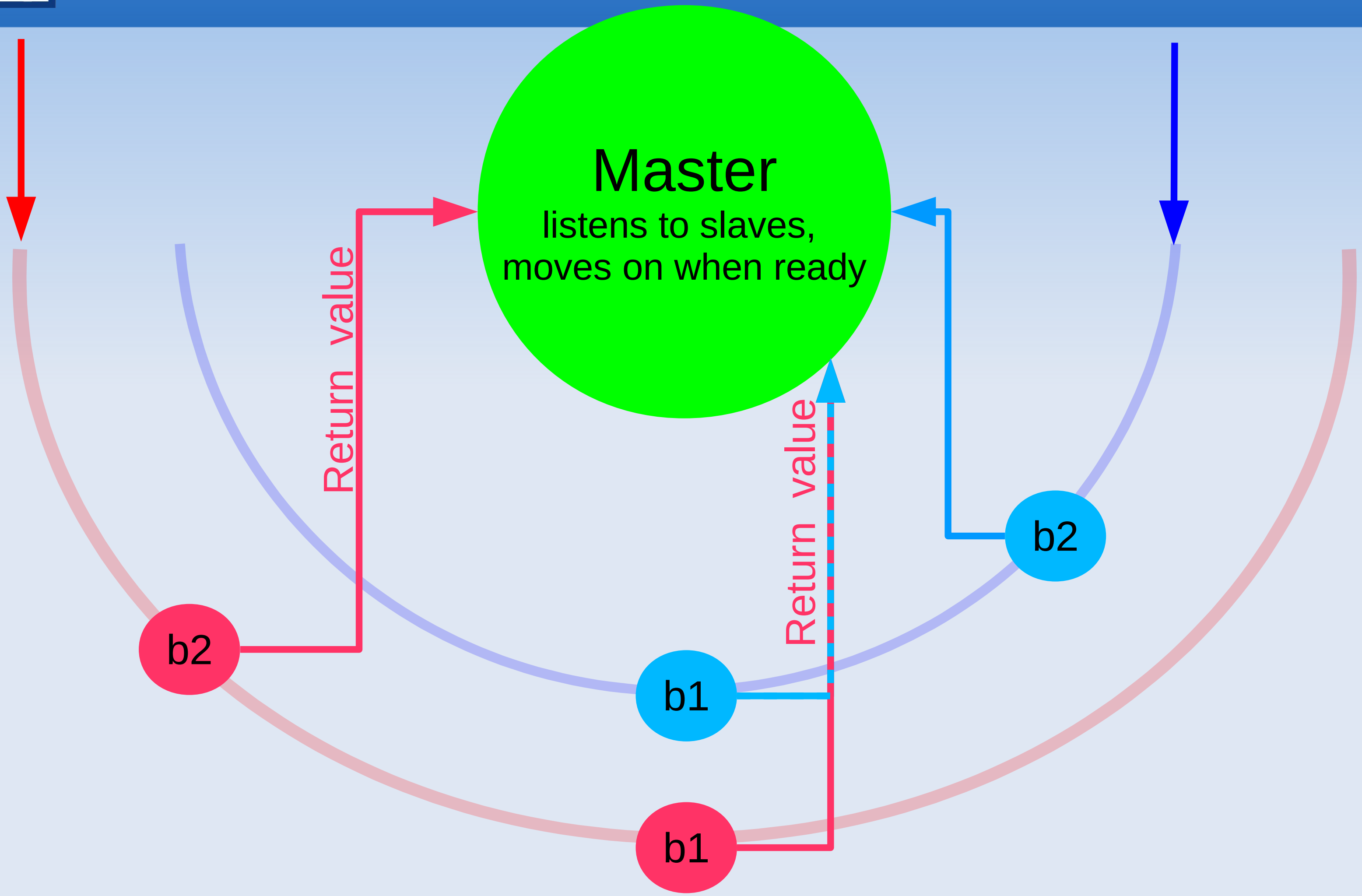
Comunication



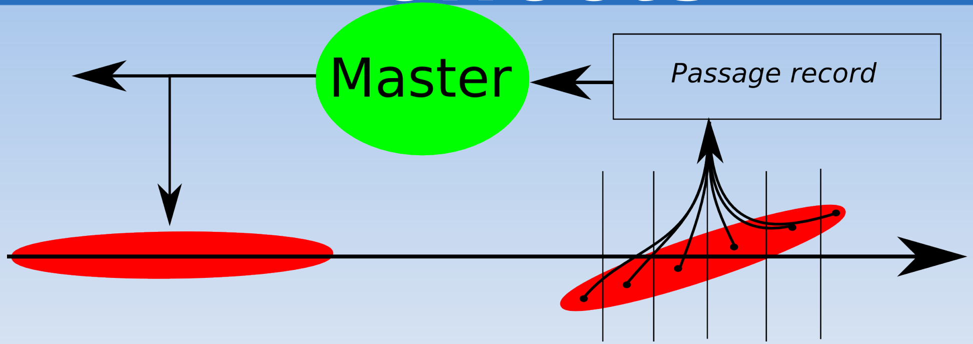
Comunication



Comunication



Multibunch / multiturn effects



- Some effects (now only the impedance → ADT features ?) requires memory of the particle distribution of the different bunches over a some turns

  → A PassageRecord instance is created evey time a bunch reaches an Impedance Action which contains :

  - The absolute time of the interaction
  - First order moments of the particle distributions (either per bunch or per longitudinal slice)

- The PassageRecord is send to the master and forwarded to the slaves when needed
  - Each slave keeps in memory its own deque with PassageRecord of all other bunches
  - Maximum length of the deque is set by the 'wake length' in input

Typical setup

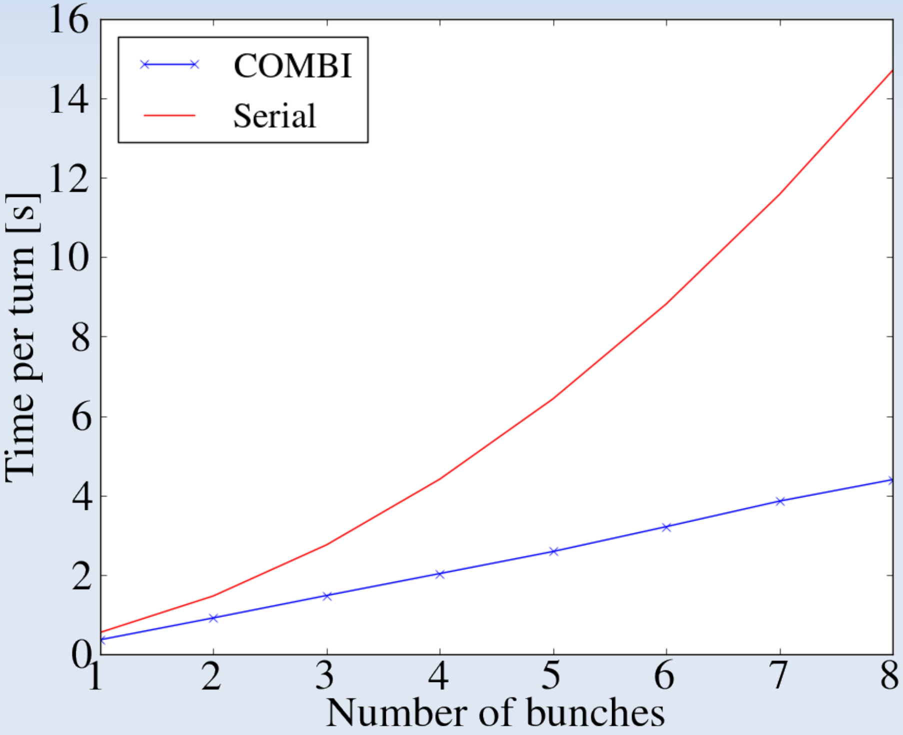
Study	Nb particle per bunch	Nb bunch per beam	Nb slice per bunch	Nb Turn	Nb of run per scan	Preferred field solver	Run time
Computation of coherent beam-beam mode spectrum	10^4-10^5	1-10^4	1	10^4	1	HFMM	Minutes to hours
Stability threshold / Mode coupling instability	10^5-10^6	1-36 (-10^4)*	50-500	10^4-10^6	10-100	Soft-Gaussian	Minutes to days
BTF	10^4-10^6	1-36 (-10^4)*	1	10^4	50-100	Soft-Gaussian / HFMM	Hours
Emittance effect / distribution effects	> 10^6	1 (-10^4)*	1	> 10^6	20	FPPS / HFMM	Days to weeks

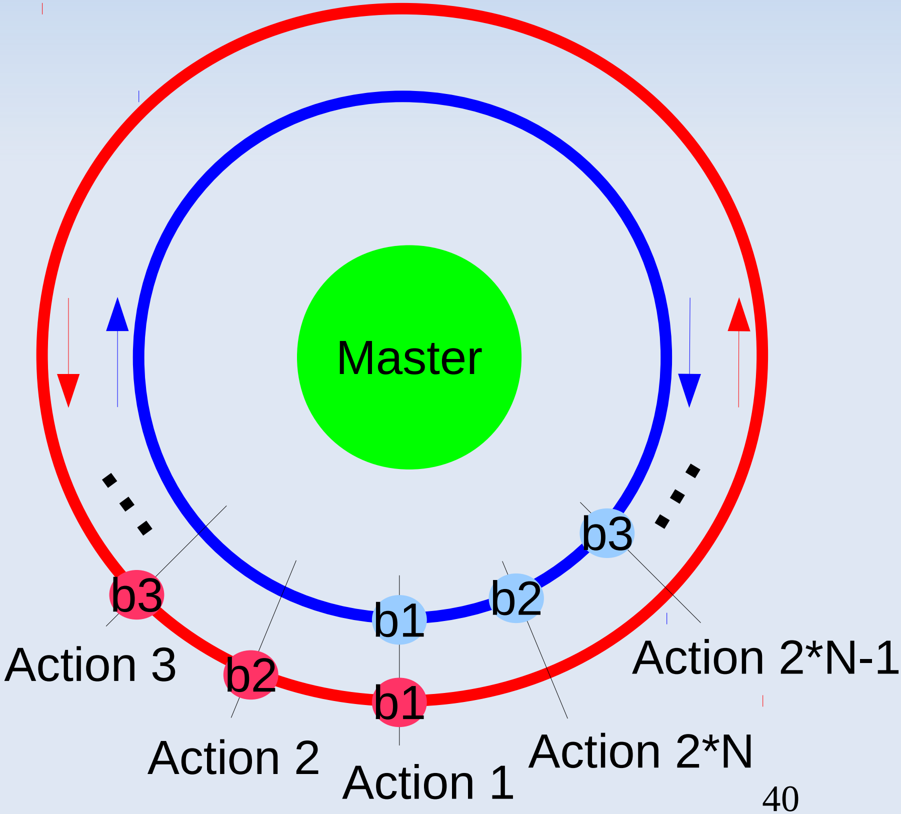
* The total number of bunches in the FCChh would be out of reach currently

First level of parallelization

Performance

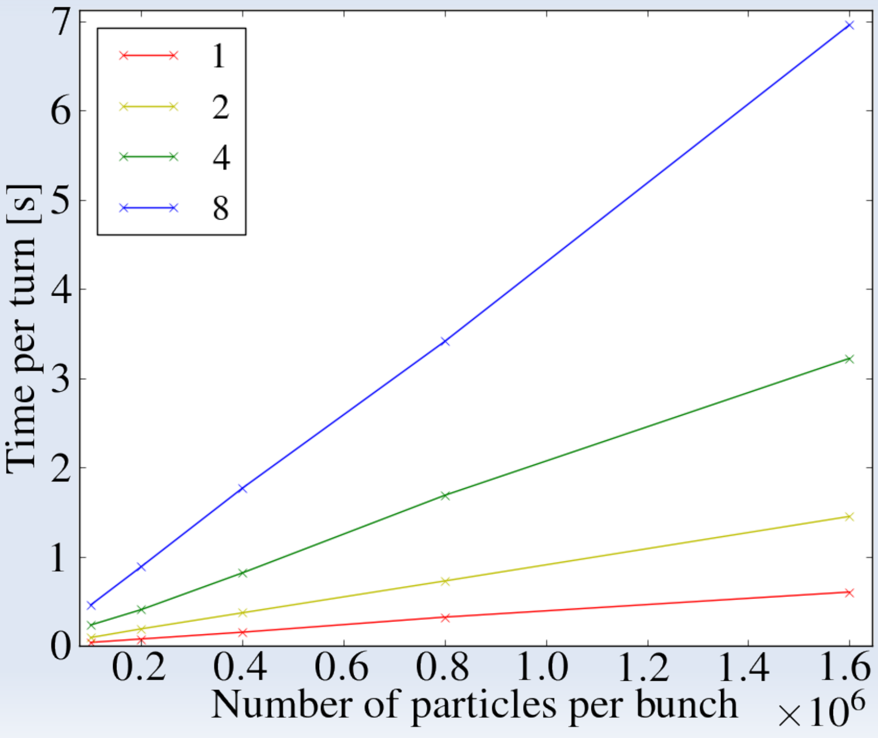
- Total number of actions to be performed ~ N_{bunch}^{2}
- Number of actions per bunch ~ N_{bunch}

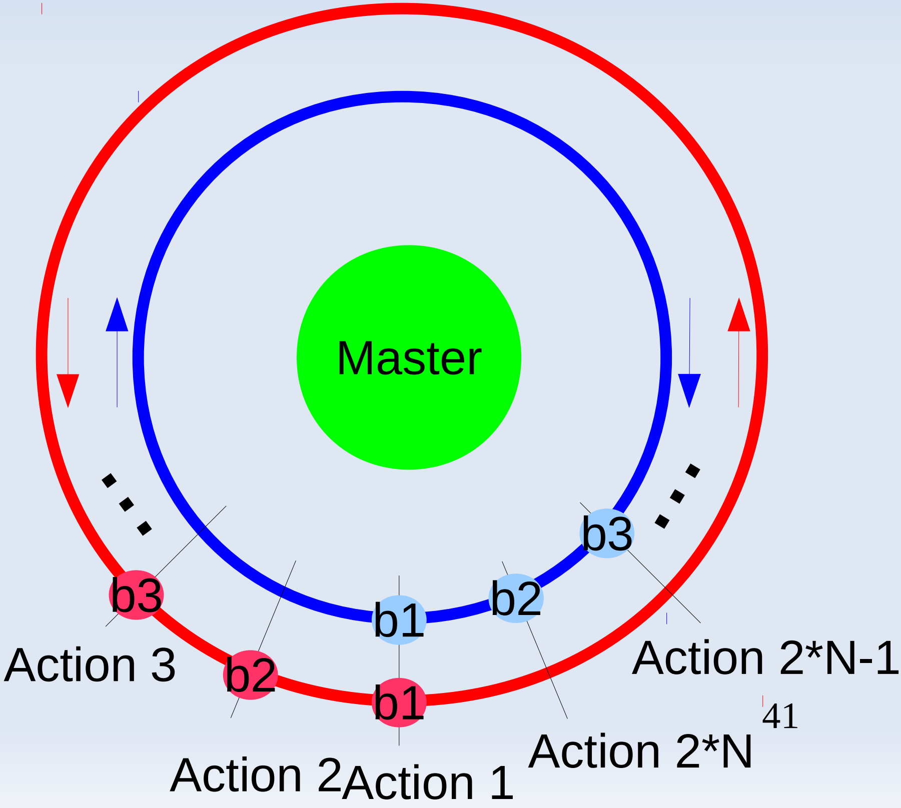




First level of parallelization
Drawbacks

- No gain for actions requiring heavy processing
- Potentially large waste of resources (waiting processes)
  - Flexibility in the action sequence is a requirement driven by the different needs





Second level of parallelization COMBI hybrid (COMBhy)

- No change on the first level of parallelization
- Parallelize loops using OpenMP

✓ *Simple implementation*

✓ *Many optimisation features available in OpenMP*

✗ *Resources are wasted because of the busy-waiting of idling MPI processes*

✗ *Control of priorities at the OS scheduler level is available in COMBI (enable RTS) but it requires privileges that are usually not granted on a shared cluster (use with care, even on your machine...)*

Second level of parallelization COMBI with shared memory (COMBIsh)

- Requires major modification of the first level of parallelization :
  - Third type of process : **helper**
- **Slaves** and **helpers** have access to shared memory (using POSIX memory mapping)

✓ *No idling MPI process*

✗ *Complicated implementation*

✗ *Work sharing algorithm have to be implemented*

Second level of parallelization
COMBI with shared memory (COMBIsh)

- Requires major modification of the first level of parallelization :
  - Third type of process : **helper**
- **Slaves** and **helpers** have access to shared memory (using POSIX memory mapping)

✓ *No idling MPI process*

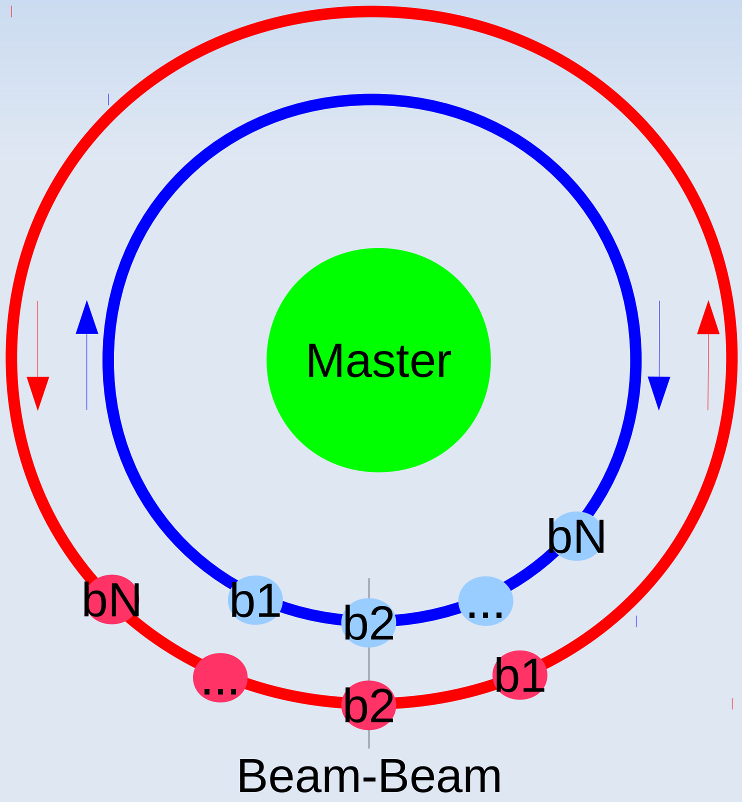
✗ *Complicated implementation*

✗ *Work sharing algorithm have to be implemented*

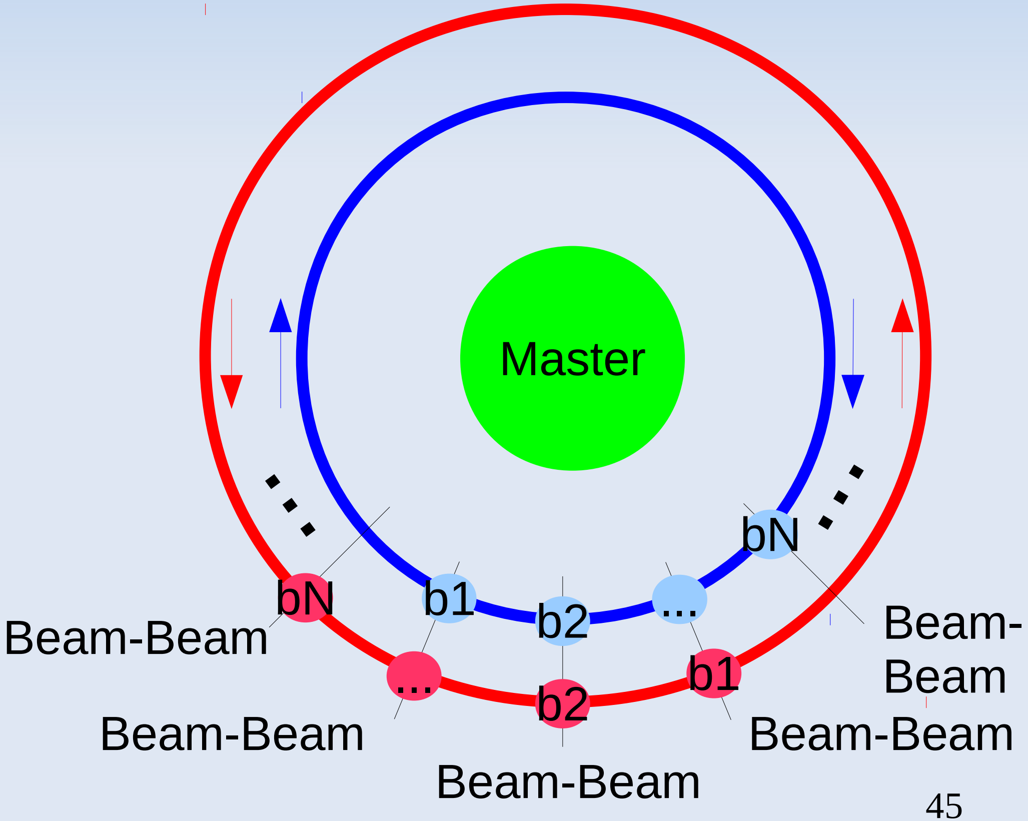
Discontinued

Performance
Study cases

- High waste (low perfromance with first level of parallelization) :



- Low waste (High perfromance with first level of parallelization) :

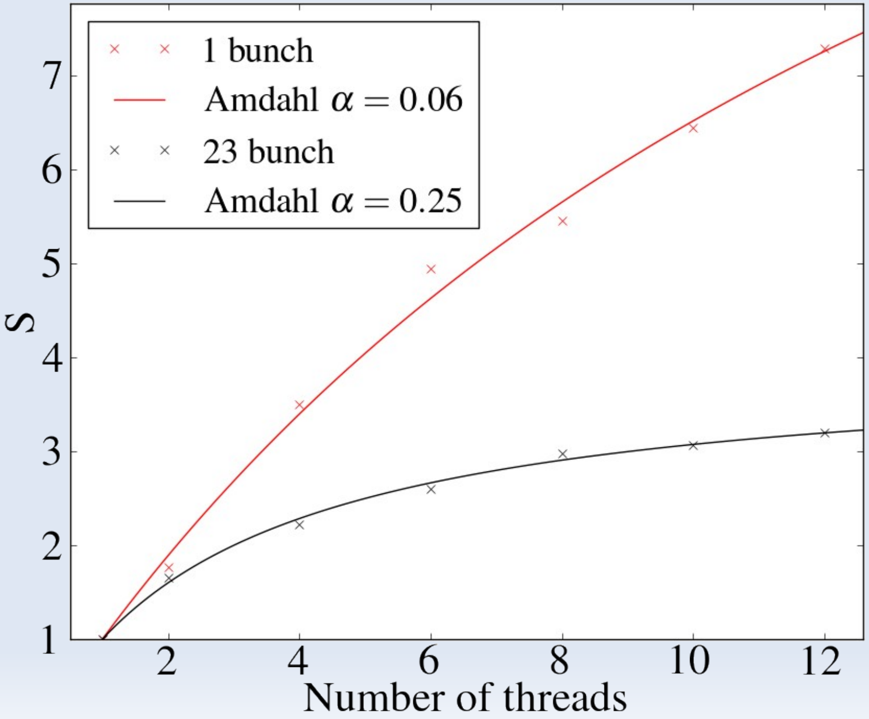


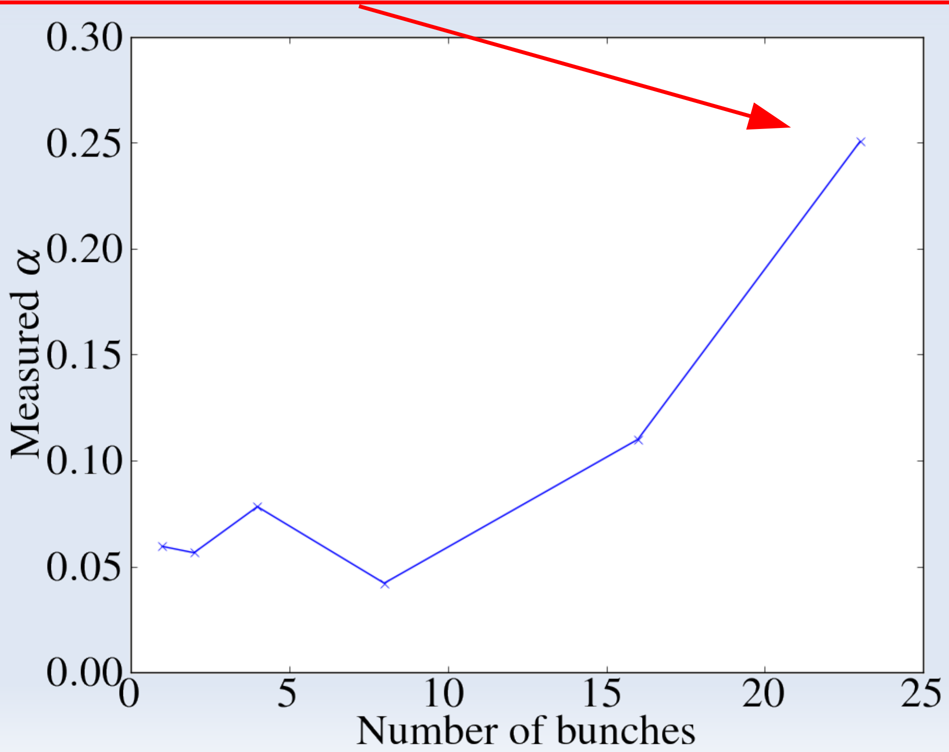
Performance
COMBhy – high waste config.

- ✔ ~95% of the code seems parallelizable

- ✗ 23 bunches → 47 MPI processes running on 48 CPUs, only 2 of which are working at a time

  → Waste of resources still present

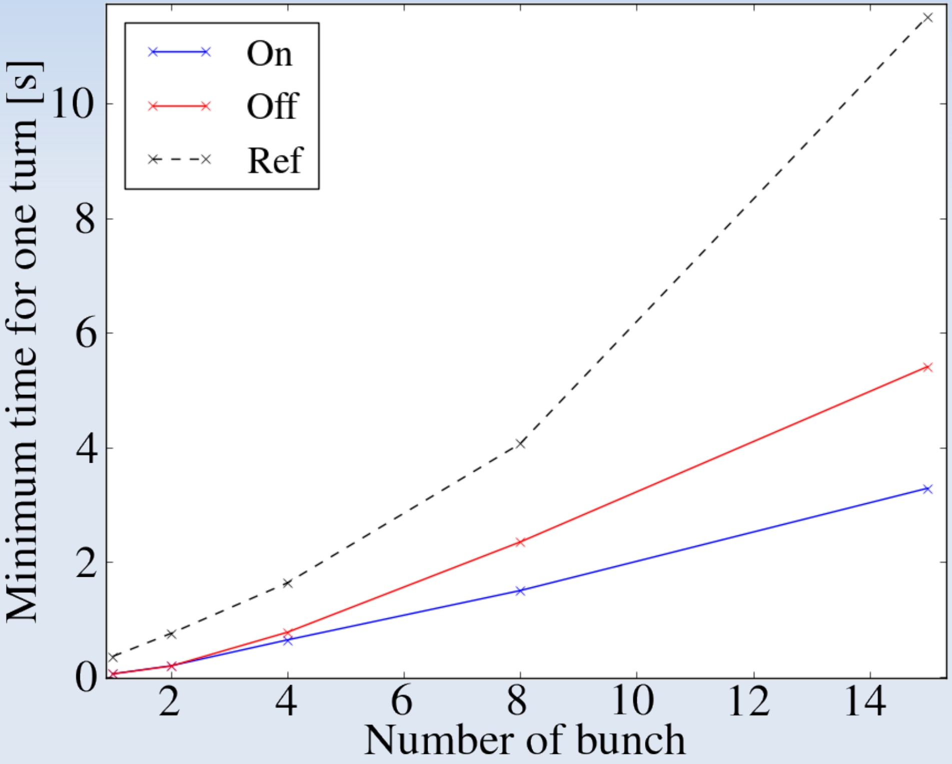




Performance
COMBhy – high waste config.

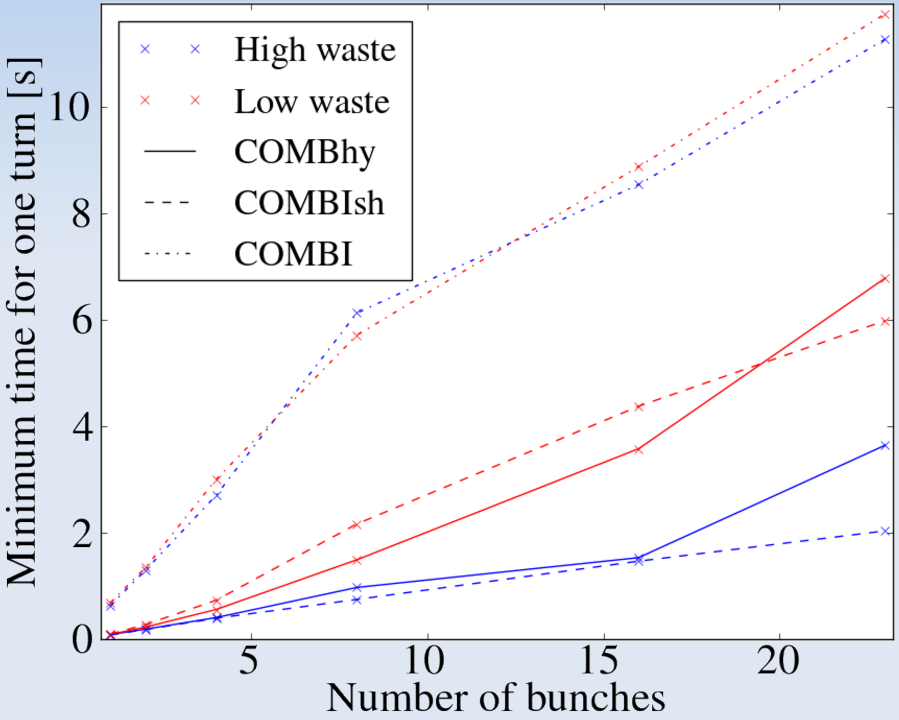
- The waste of resource can be mitigated by introducing system calls that to lower the priority of idling MPI process with respect to processing ones in the OS scheduler
- The gain is marginal in absence of oversubsribtion but otherwise significant
- Changing processes priorities requires capabilities that are not always granted

16 core HP blade 2.7 GHz with HT



Performance

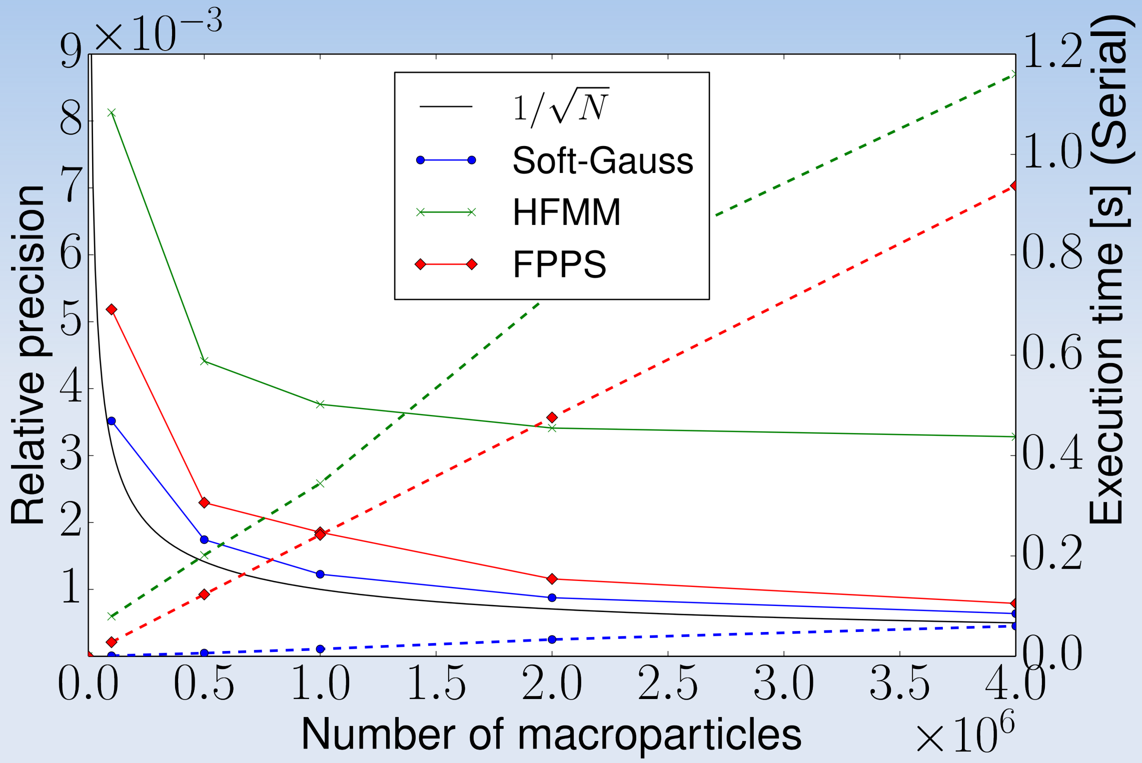
- The second level of paralelisation offers a major speedup wrt to the single level only, both in the high and low waste configurations
- The shared memory version does not allow for a major speedup in the high waste configuration



Solver performance

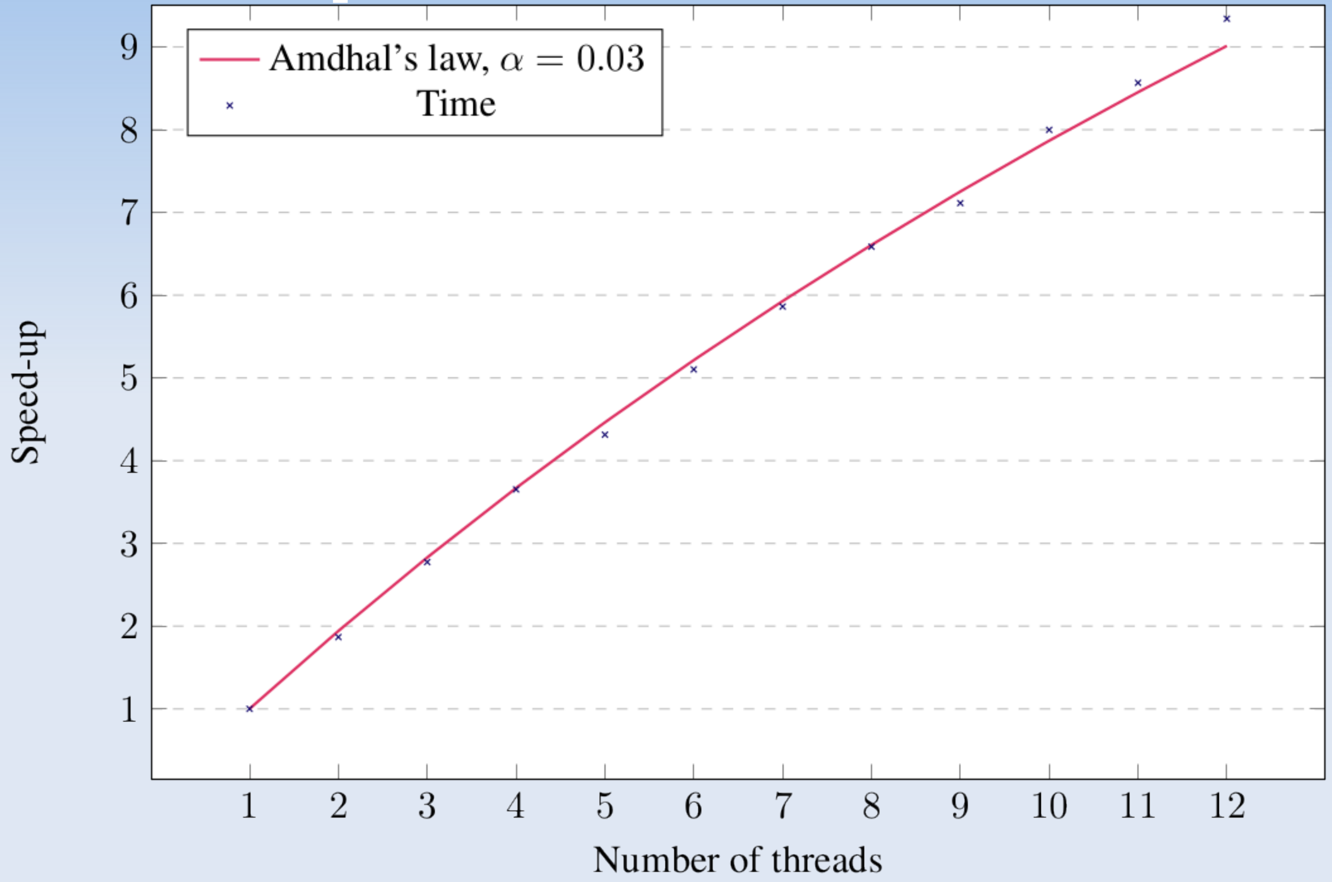
- The noise introduced by the field solver is critical, especially when studying slow emittance effects
- Estimated by executing it on a random distribution for which an analytical formula exists
  - Average of several seeds

$$\Delta_{solver} = \frac{\langle |k_{solver} - k_{th}| \rangle}{\langle |k_{th}| \rangle}$$



- The best performing is the soft-Gaussian as it is fast and its noise approaches the theoretical minimum 1/sqrt(N)
- The FPPS offers a major speed up for a given noise amplitude wrt to the HFMM

FPPS parallelisation performance



- Excellent speedup using OpenMP on the most of the loops (not within fftw)

Performance vs needs

- Many studies are run on local machines (usually 4 to 12 cores), which performances allow for decent studies
- For most studies the code performs well, but requires proper parallel infrastructure, as well as manpower for setting up, testing and optimisation on each architecture

  → Preferred resources : Multiple nodes with fast connection, regular memory and possibly several cores per node
- EPFL infrastructures (accessible on request by L. Rivkin's Ph D. students) offer such capabilities, most results obtained with COMBI where based on their facilities
- Need to investigate potential of current CERN infrastructures

Future plans

- Current implementation has great potential
  - Maintain and use the code to produce results (including implementation of 'minor' actions for interplay studies)
- Emittance studies are limited by the amount of numerical noise due to the field solver with a finite number of macro-particles (10^6 → 10^7)
  - GPU acceleration is not suited for strong-strong simulations due to the communication requirements (imposed by the physics)

    → Need for computing resources (large number of CPUs per node)
- Long term stability studies (either due to slow rise times, or slow distorsions of the particle distribution) requires several turns (~10^6 or more) and several parameter scans

    → Need for computing resources (large number of CPUs per node, possibly large number of nodes for multibunch studies)
- Effort on the 6D beam-beam effects just (re-)started

    → Effectively increases the computational need by a factor ~50 (number of slices per bunch)
  - Testing and parallelisation of the 6D soft-Gaussian solver
  - Implementation of a self-consistent 6D solver ?

Documentation

- Source are available on svn : https://svnweb.cern.ch/cern/wsvn/COMBI
  - doc/action codes
  - doc/input conventions
  - doc/goottoknow
- https://cds.cern.ch/record/1987672/files/CERN-THESIS-2014-246.pdf appendix A
- https://twiki.cern.ch/twiki/bin/view/ABPComputing/COMBI

… better than nothing

License

- CERN copyright

References

- http://wwwslap.cern.ch/collective/hirata/
- J. Carrier, L. Greengard, V. Rokhlin, A Fast Adaptive Algorithm for Particle Simulation, Yale U. Comp. Sci. Dept. RR # 496. Sep.86, Revised Jan.87
- F.W. Jones, H.O. Schönauer, New Space-Charge Methods in Accsim and Their Application to Injection in the CERN PS Booster, 1999 Particle Accelerator Conference, New York, USA
- W. Herr, M.P. Zorzano, F. Jones, A hybrid fast multipole method applied to beam-beam collisions in the strong-strong regime, Workshop on Beam-beam Effects, Fermilab, Batavia, IL, USA , 25 - 28 Jun 2001
- W. Herr and F. Jones, Parallel computation of beam-beam interactions including longitudinal motion, PAC 2003, Portland, US
- T. Pieloni, W. Herr, Models to Study Multi-bunch Coupling through Head-on and Long-range Beam-beam Interactions, EPAC 2006, Edinburgh, Scotland
- F. Jones, W. Herr, T. Pieloni, “Parallel Beam-Beam Simulation Incorporating Multiple Bunches and Multiple Interaction Regions”, Proceedings of the 22nd Particle Accelerator Conference, Albuquerque, New Mexico, USA (2007)
- T. Pieloni, A Study of Beam-Beam Effects in Hadron Colliders with a Large Number of Bunches, PhD thesis, EPFL, 2008
- F. Jones, W. Herr, T. Pieloni, “Self-Consistent Parallel Multi Bunch Beam-Beam Simulation using a Grid-Multipole Method”, Proceedings of the 23rd Particle Accelerator Conference, Vancouver, Canada (2009)
- X. Buffat, Transverse Beams Stability Studies at the Large Hadron Collider, PhD thesis, EPFL, 2015
- A. Florio, X. Buffat, T. Pieloni, Fast Poisson Solver for self-consistent beam-beam and space-charge field computation in multiparticle tracking simulations, CERN-ACC-NOTE-2015-0038