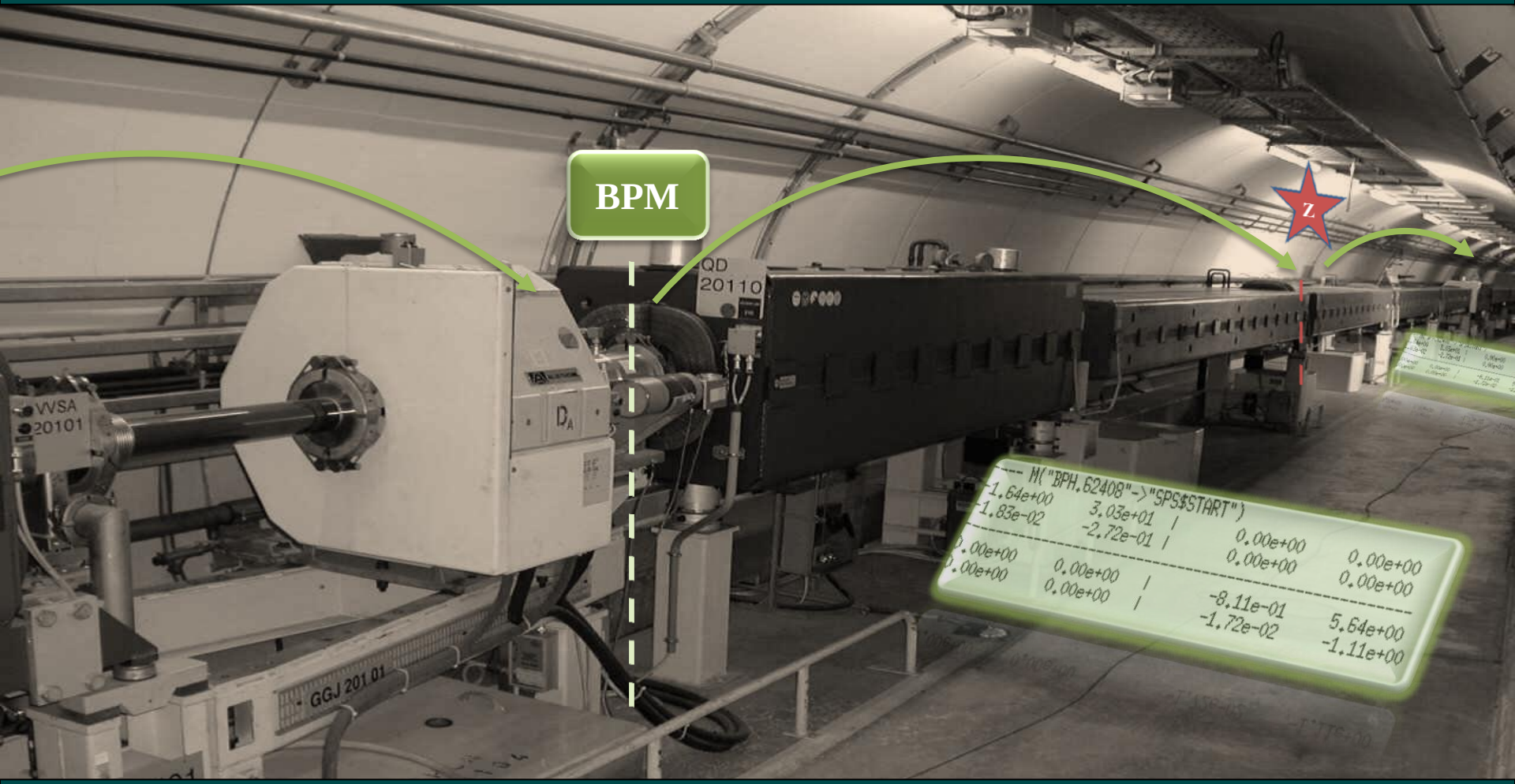
HEADTAIL + MAD-X

N.Biancacci



Acknowledgement: B.Salvant, E.Métral, N.Mounet, D.Quatraro, G.Rumolo.

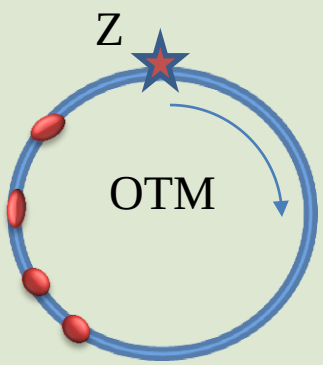
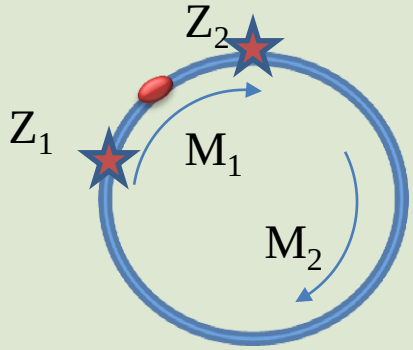
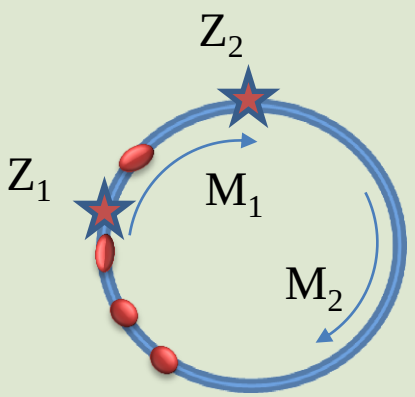
Outline

PART I:

- Present versions for HEADTAIL,
- HDTL+MADX: the general approach,
- Wakefield management,
- Lattice management,
- HEADTAIL,
- Output management.

PART II: Application to the damping ring of CLIC (E. Koukovini Platia).

Present versions of HEADTAIL

Single kick Multi bunch Multi turn (Nicolas ver.)	Multi kick Single bunch (Diego's ver., see Diego's PhD thesis)	Multi kick Multi bunch Multi turn
		
All the impedances are lumped in the same point. The beam is tracked with a One Turn Map and each turn the interaction with the global impedance Z is calculated. Possibility of multiple bunches.	The beam is tracked along the MAD-X lattice creating matrices M_1, M_2.. M_i between the points where the Z_i interaction is calculated.	Same philosophy as Diego's ver. but very different implementation of the lattice to/from MAD-X. Possibility of multi bunches as implemented by Nicolas.

General approach

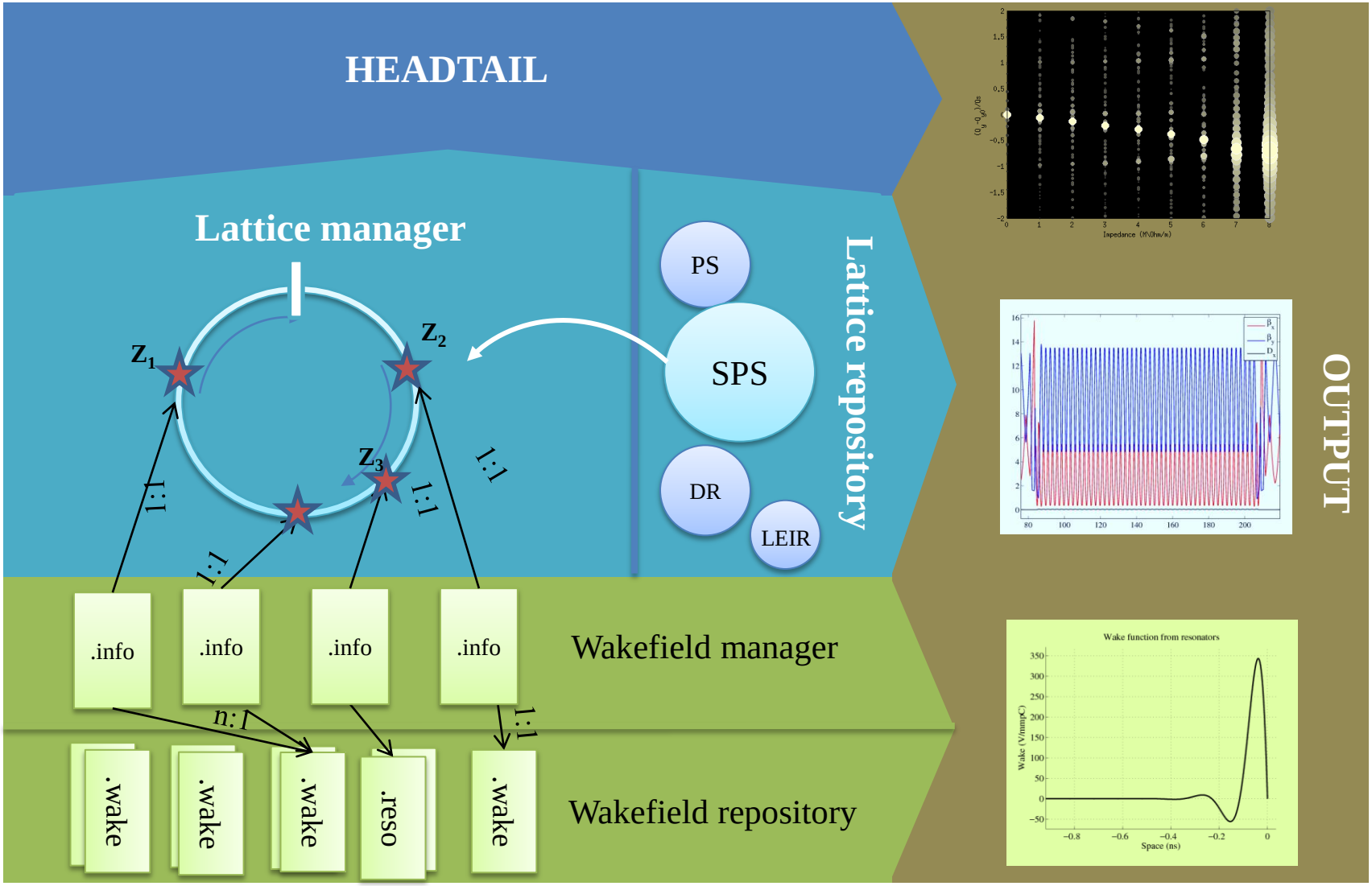
The idea…

The idea is to give a *"backbone"* to HEADTAIL developing a MADX interface in order to:

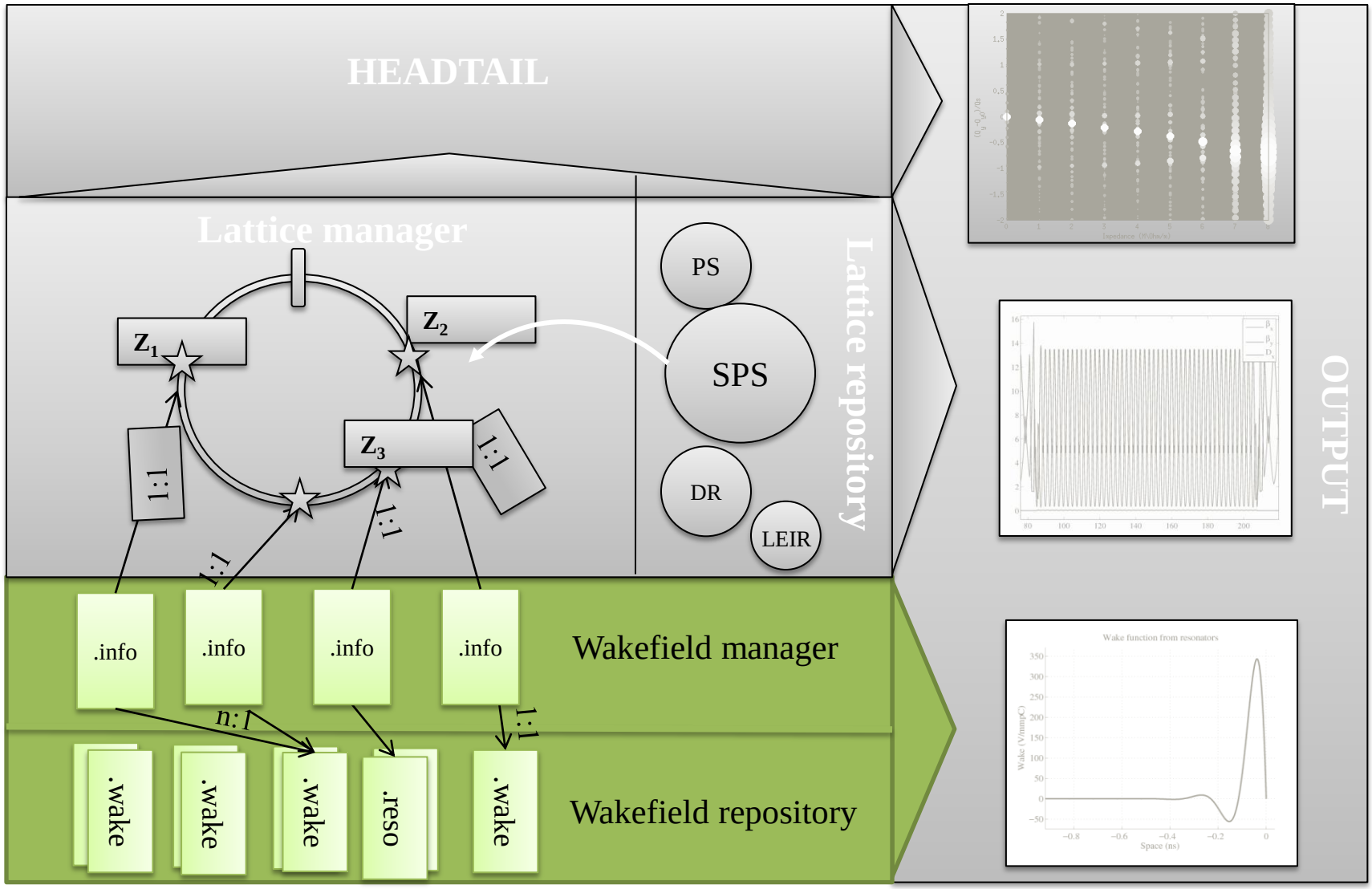
1. Make easy the extension of our studies to other machines (SPS, PS, CLIC, LEIR, damping ring (DR), etc…).
2. Make easy lumping or distributing impedances for different lattices.
3. Decrease the "entropy" keeping the code modular.
4. Improve the usability from a user point of view.

What came out…

General approach



Wakefield management



Wakefield management

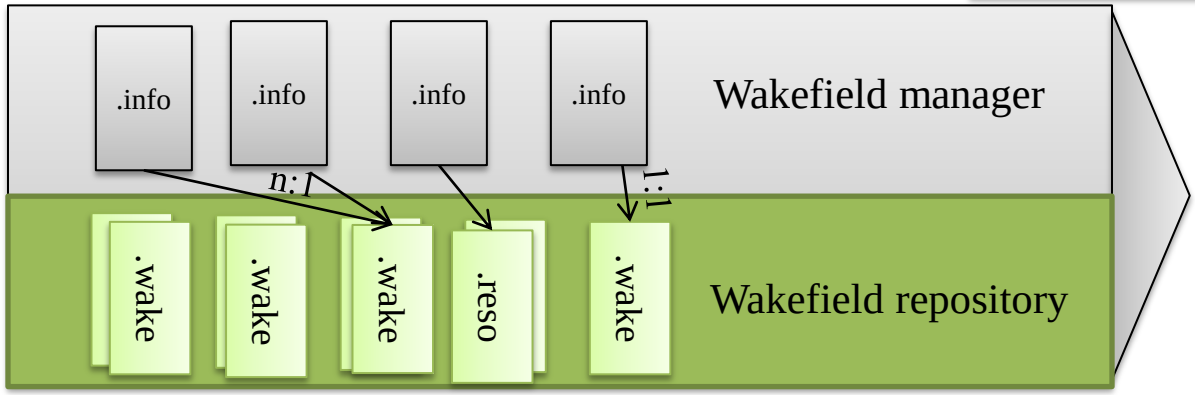
- The wakefields are collected in a specific folder named "HDTL_wakes" (see later) as raw data. For the moment two kind of impedances are present:
  1. Resonator impedances: specified in a ".reso" file.
  2. Wake tables: specified as a ".wake" file. Here the impedance is tabulated in columns accordingly to Nicolas convention (see later) .

.reso

Res_frequency_of_broad_band_resonator_[GHz]:	5.0
Transverse_quality_factor:	1.
Transverse_shunt_impedance_[MOhm/m]:	20
Res_frequency_of_longitudinal_resonator_[MHz]:	200.
Longitudinal_quality_factor:	140.
Longitudinal_shunt_impedance_[MOhm]:	0.0

.wake

Time (ns)	Wxdip	Wydip	Wxquad	Wyquad
0	0.72349	1.1619	-0.72562	0.80356
0.016678	0.87796	1.37	-0.88122	0.99735
0.033356	1.021	1.5424	-1.0257	1.1878
0.050035	1.1409	1.6672	-1.1482	1.3548
0.066713	1.2294	1.7368	-1.2399	1.4798
0.083391	1.2819	1.7504	-1.2959	1.5515
0.10007	1.2966	1.7113	-1.3158	1.5679
0.11675	1.2766	1.6259	-1.3022	1.5344
0.13343	1.2275	1.5032	-1.2603	1.4614



Wakefield management

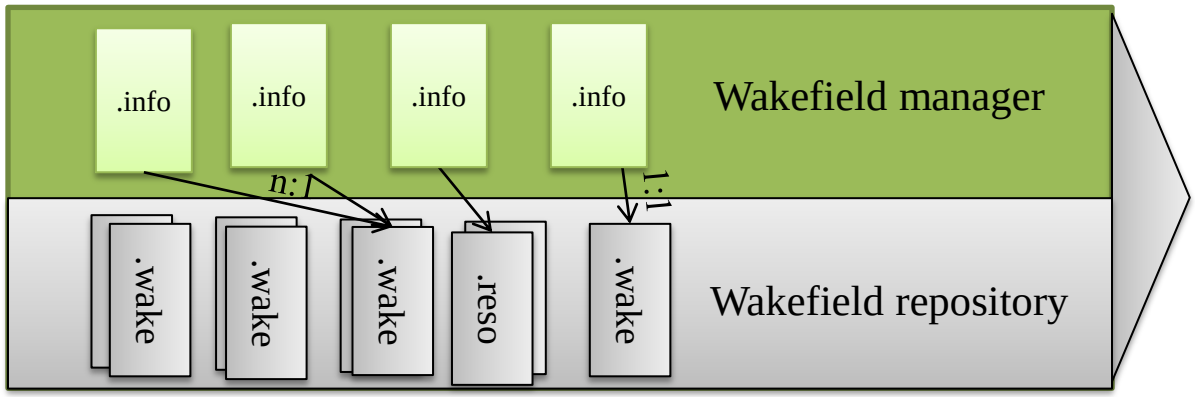
- In order to be able to run simulations with different impedances, of whatever nature, in whatever location, a *"Wakefield manager"* will read a ".info" file that contains information about the impedance we want to place in the lattice and simulate.

.info to manage a resonator

NAME:	BB_SPS
POSITION:	100
INTERACTION:	IMPEDANCE
TYPE:	0
SCALE:	0
BETX:	-
BETY:	-
MULTIPLY:	0
MULTIPLY_COEFF:	1

.info to manage a wake table

NAME:	MKE.61637
POSITION:	6281.80
INTERACTION:	IMPEDANCE
TYPE:	1
TABLE_TYPE:	4
SCALE:	0
BETX:	-
BETY:	-
MULTIPLY:	0
MULTIPLY_COEFF:	1



NB: the name of the ".info" file is the name of the element present or installed in the MADX lattice. The name in the ".info" file is the one pointing to the ".wake" one. For example we have an element RFCAVITY in the lattice, then we have a RFCAVITY.info file that points to rf200Mhz.wake (NAME=rf200MHz)

Wakefield management

- In order to be able to run simulations with different impedances, of whatever nature, in whatever location, a *"Wakefield manager"* will read a ".info" file that contains information about the impedance we want to place in the lattice and simulate.

.info to manage a resonator

NAME:	BB_SPS
POSITION:	100
INTERACTION:	IMPEDANCE
TYPE:	0
SCALE:	0
BETX:	-
BETY:	-
MULTIPLY:	0
MULTIPLY_COEFF:	1

.info to manage a wake table

NAME:	MKE.61637
POSITION:	6281.80
INTERACTION:	IMPEDANCE
TYPE:	1
TABLE_TYPE:	4
SCALE:	0
BETX:	-
BETY:	-
MULTIPLY:	0
MULTIPLY_COEFF:	1

NAME	specify the ".reso" or ".wake" file you want to point to. This will be then plugged in the lattice. It is possible to have different ".info" file pointing on the same wakefield.
POSITION	specify the position in which the impedance will be inserted in the lattice if not already present (see later).
INTERACTION	specify the kind of interaction to be done by HEADTAIL. In future could be a ecloud or space charge interaction…
TYPE	if 0 the name will be referred to a ".reso" file, if 1 to a ".wake" file. It is the *"i_pipe"* flag in the other version. The other impedances, case-switched in the past versions, can still be simulated accordingly to their old *"i_pipe"* flag. NB: The case 1 corresponds to case 8 of Nicolas's version.
TABLE TYPE	Only for wake tables (".wake"). Specify the wakes you are going to simulate (see next slide).
SCALE	if 1 enable scaling of transverse wake fields (only dipolar and quadrupolar ones) in order to match the target beta function BETX and BETY(see later for more about this). If 0 the wake will be applied accordingly to the beta functions at the place the kick is given.
BETX, BETY	Horizontal and vertical target beta function;
MULTIPLY	if 1 enable scaling of transverse wake fields of a factor given by MULTIPLY_COEFF. Useful if you want to split the same impedance along different position in the ring.

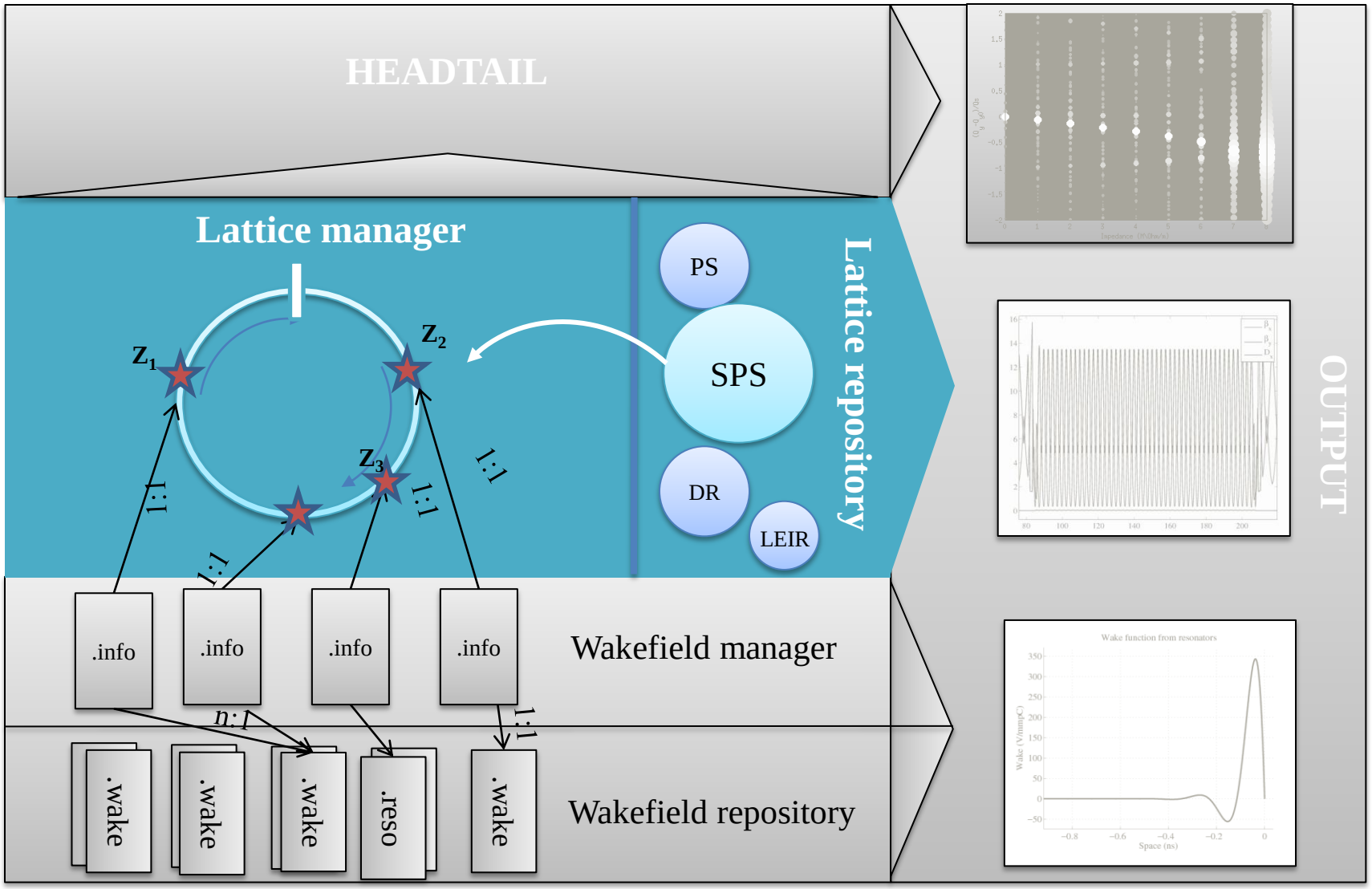
Wakefield management

WAKE TABLE

Wake table are taken accordingly to the convention that the most general wakefield is made by dipolar, quadrupolar, coupled, constant and longitudinal components. When reading a wake table (TYPE=1) the user have to specify with TABLE_TYPE the wakes that are contained in the corresponding ".wake" file.

TABLE_TYPE	Columns in ".wake" file
1	T[ns], Wlong[V/pC].
2	T[ns], Wxdip [V/(mm.pC)], Wydip [V/(mm.pC)].
3	T[ns], Wxdip [V/(mm.pC)], Wydip [V/(mm.pC)], Wlong[V/pC].
4	T[ns], Wxdip [V/(mm.pC)], Wydip [V/(mm.pC)], Wxquad [V/(mm.pC)], Wyquad [V/(mm.pC)].
5	T[ns], Wxdip [V/(mm.pC)], Wydip [V/(mm.pC)], Wxquad [V/(mm.pC)], Wyquad [V/(mm.pC)], Wlong[V/pC].
6	T[ns], Wxdip [V/(mm.pC)], Wydip [V/(mm.pC)], Wxquad [V/(mm.pC)], Wyquad [V/(mm.pC)], Wxydip[V/(mm.pC)], Wxyquad[V/(mm.pC)].
7	T[ns], Wxdip [V/(mm.pC)], Wydip [V/(mm.pC)], Wxquad [V/(mm.pC)], Wyquad [V/(mm.pC)], Wxydip[V/(mm.pC)], Wxyquad[V/(mm.pC)], Wlong[V/pC].
8	T[ns], Wxdip [V/(mm.pC)], Wydip [V/(mm.pC)], Wxquad [V/(mm.pC)], Wyquad [V/(mm.pC)], Wxydip[V/(mm.pC)], Wxyquad[V/(mm.pC)], Wxconst[V/pC], Wyconst[V/pC].
9	T[ns], Wxdip [V/(mm.pC)], Wydip [V/(mm.pC)], Wxquad [V/(mm.pC)], Wyquad [V/(mm.pC)], Wxydip[V/(mm.pC)], Wxyquad[V/(mm.pC)], Wxconst[V/pC], Wyconst[V/pC], Wlong[V/pC].

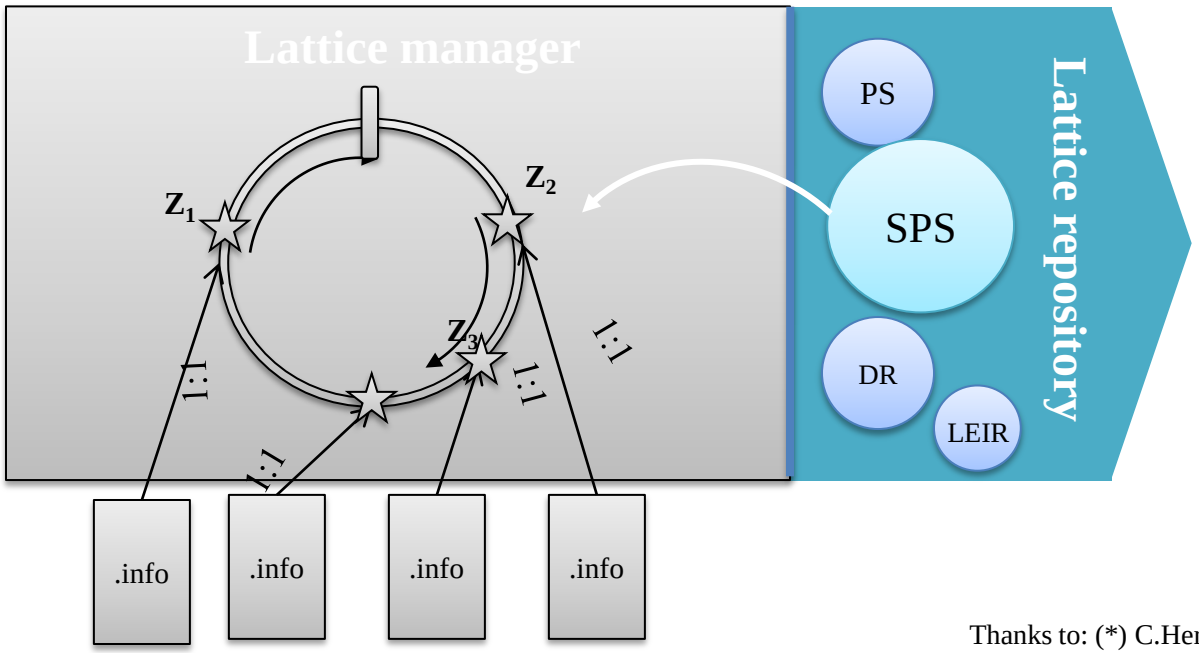
Lattice management



Lattice repository

HEADTAIL is now interfaced with MAD-X lattices. These are kept inside a specific folder with the name of the machine. The machines present until now are:

- SPS
- PS*
- DR (CLIC damping ring)**
- LEIR***



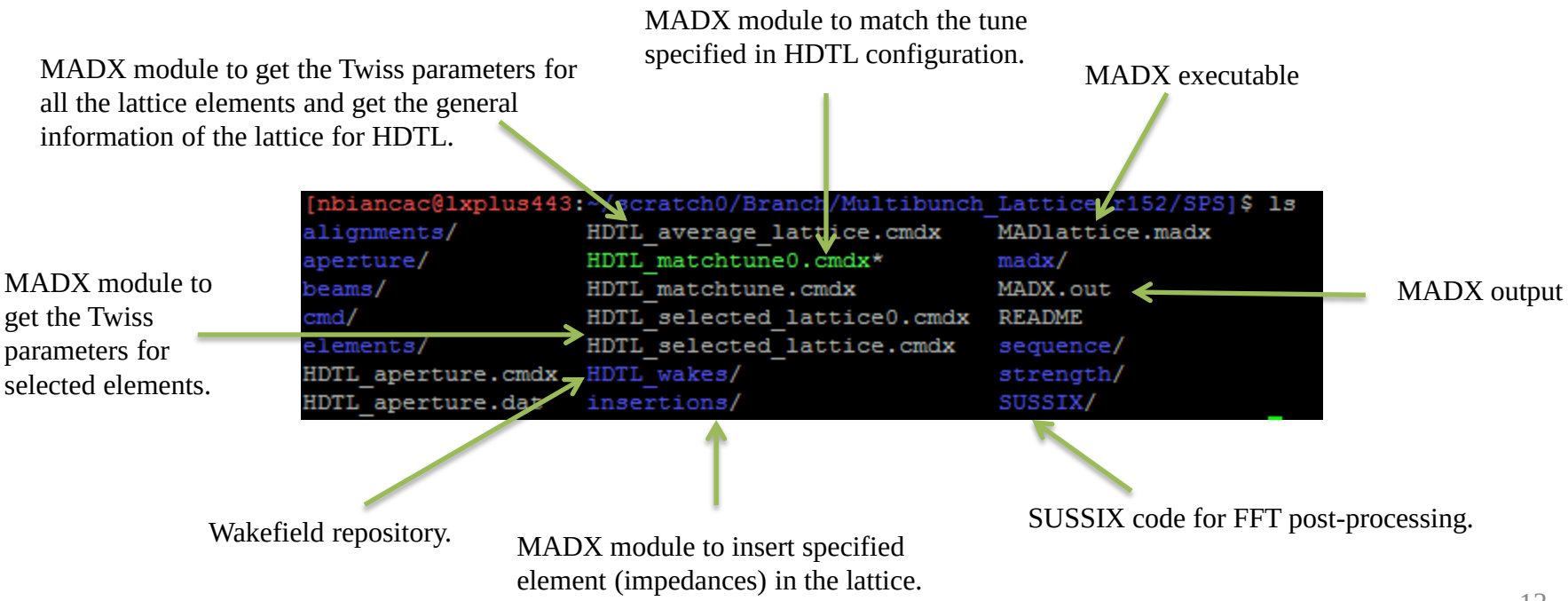
Thanks to: (*) C.Hernalsteens, (**) F.Antoniou, (***) O.Berrig.

Lattice repository

HEADTAIL is now interfaced with MAD-X lattices. These are kept inside a specific folder with the name of the machine. The machines present until now are:

- SPS
- PS
- DR (CLIC damping ring)
- LEIR

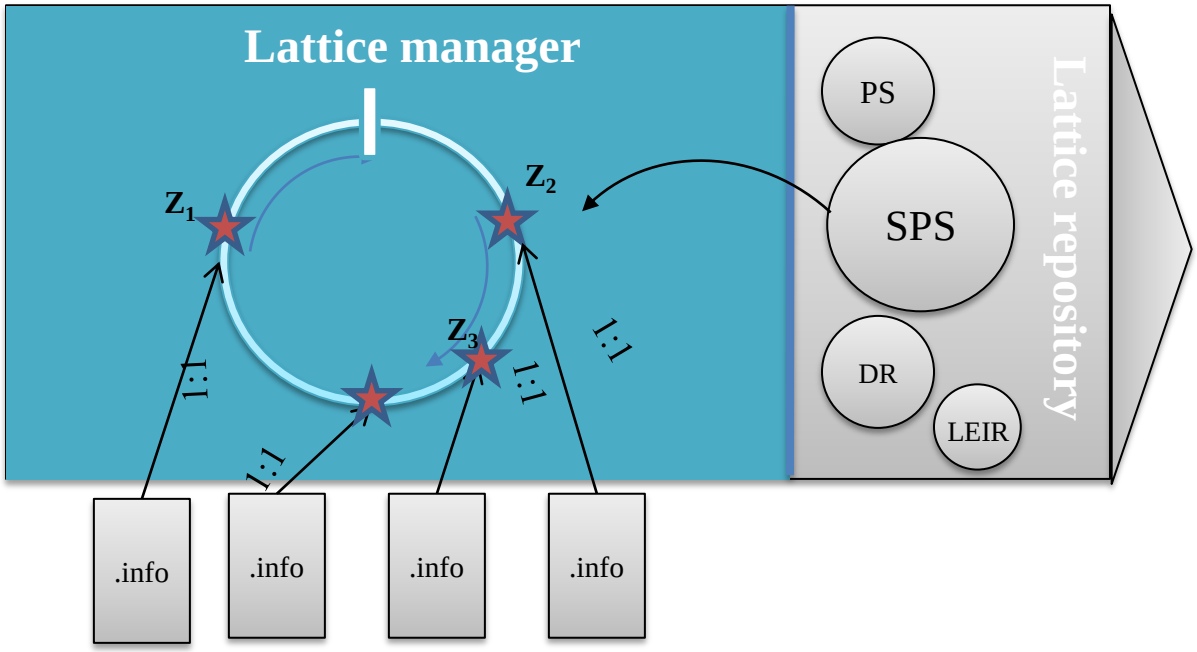
It looks something like this…



Lattice manager

Once a machine is specified, HEADTAIL will do the following operations:

1. Read the ".info" files specified in the configuration file.
2. Place the impedances along the lattice and, if the case, install new element in the lattice.
3. Make a struct of each element to handle easily the twiss parameters.
4. Construct the transport matrices from one element to the other.
5. If the impedances have to be lumped at one point construct and use the One Turn Map.



Lattice manager - elements

Until now three type of **elements** are present:

- **START**: always present, is the starting point of the machine (note that the twiss alfa parameter can be different from zero here).
- **H/VMONITOR**: correspond to BPMs specified in the MADX lattice.
- **IMPEDANCE**: correspond to a wakefield interaction point.

Every element in the lattice is treated as a structure-object. It creates a structure in HEADTAIL with all its optic functions.

keyword="**MARKER**"
name="**SPS$START**"
s=0.000000e+00
mux=0.000000e+00
muy=0.000000e+00
betx=1.034789e+02
bety=2.089947e+01
alfx=-2.316795e+00
alfy=5.375344e-01
Dx*=1.231801e+00
Dy*=0.000000e+00

keyword="**VMONITOR**"
name="**BPV.10108**"
s=3.176520e+01
mux=1.094653e-01
muy=1.307504e-01
betx=2.121867e+01
bety=1.020872e+02
alfx=5.522338e-01
alfy=-2.295629e+00
Dx*=9.463603e-01
Dy*=0.000000e+00

keyword="**IMPEDANCE**"
name="**MKE.41631**"
s=3.972341e+03
mux=1.500822e+01
muy=1.506461e+01
betx=9.637988e+01
bety=2.270612e+01
alfx=2.218481e+00
alfy=-6.313039e-01
Dx*=-1.573395e-01
Dy*=0.000000e+00

Moreover, elements contain pointers to FILE that are going to be written or read.

- START struct contains a pointer to the "prt.dat" file for back compatibility.
- H/VMONITOR structs contain pointers to their own output file (see later).
- IMPEDANCE structs contain pointers to their own input wake field and output ".track" file (see later).

NB: the dispersion is corrected by a factor β (relativistic) in HEADTAIL. This is because in MADX the dispersion is referred to a momentum spread given by $\delta_{MAD} = \frac{\Delta E}{p_0 c} = \beta \frac{\Delta p}{p_0} = \beta \delta_{HEADTAIL}$

Lattice manager - transport matrices

Once the lattice is created, each element has a 4-D **transport matrix** associated from its point to the following one.

---------- Matrix of elements ----------

M("SPS$START"->"BPV.10108)

-3.16e-01	2.97e+01	0.00e+00	0.00e+00
-4.35e-02	9.32e-01	0.00e+00	0.00e+00
0.00e+00	0.00e+00	2.38e+00	3.38e+01
0.00e+00	0.00e+00	4.55e-02	1.07e+00

M("BPV.10108->"BPH.10208)

2.38e+00	3.44e+01	0.00e+00	0.00e+00
4.56e-02	1.08e+00	0.00e+00	0.00e+00
0.00e+00	0.00e+00	-3.17e-01	2.97e+01
0.00e+00	0.00e+00	-4.35e-02	9.21e-01

Then the One Turn Map (OTM) is calculated.

----------OTM----------

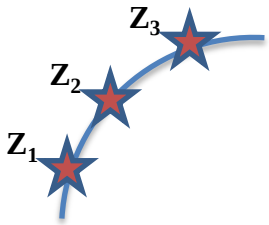
-1.00e+00	7.54e+01	0.00e+00	0.00e+00
-4.49e-02	2.37e+00	0.00e+00	0.00e+00
0.00e+00	0.00e+00	9.12e-01	1.89e+01
0.00e+00	0.00e+00	-5.58e-02	-6.06e-02

Note: Synchrotron motion, chromaticity and also octupole detuning are treated as a rotation at the end of each turn.

Lattice manager – beta functions

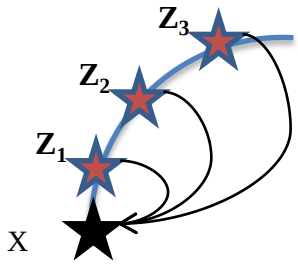
To simulate wakefield interactions, 3 approaches can be chosen accordingly to the situation.

(A) We simulate impedances one by one, the tune shift will be:



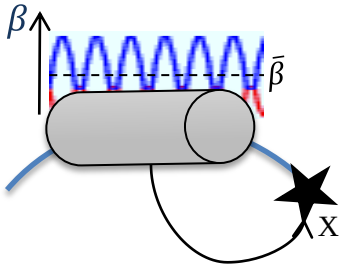
$$\Delta Q \propto \sum_k^N W_k \beta_k$$

(B) We lump all the impedances in a X point. In order to have the same effect we have to scale the wakes as:



$$\Delta Q \propto \sum_k^N W_k \beta_k = \beta_X \underbrace{\sum_k^N W_k \frac{\beta_k}{\beta_X}}_{W_X} = \beta_X W_X$$

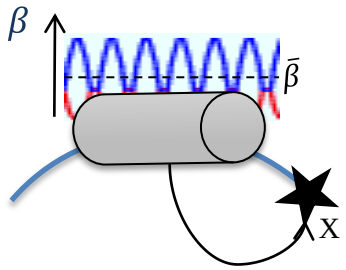
(C) In some case the impedance is smoothly distributed in long sections where the beta functions vary periodically (for example a resistive wall). We can still lump the impedance in some point X accordingly to:



$$\Delta Q \propto \oint_0^L W\beta(s)ds = WL\bar{\beta} = \beta_X \underbrace{\left(\frac{\bar{\beta}}{\beta_X} WL\right)}_{W_X}$$

Lattice manager – beta functions

(C) In some case the impedance is smoothly distributed in long sections where the beta functions vary periodically (for example a resistive wall). We can still lump the impedance in some point X accordingly to:



$$\Delta Q \propto \oint_0^L W\beta(s)ds = WL\bar{\beta} = \beta_X \underbrace{\left(\frac{\bar{\beta}}{\beta_X} WL\right)}_{W_X}$$

- With this approach we can simulate a distributed element lumping it somewhere, not necessarily at the starting point of the lattice as before.
- The $\bar{\beta}$ parameter has to be set in the ".info" file. For example:

.info to manage a wake table

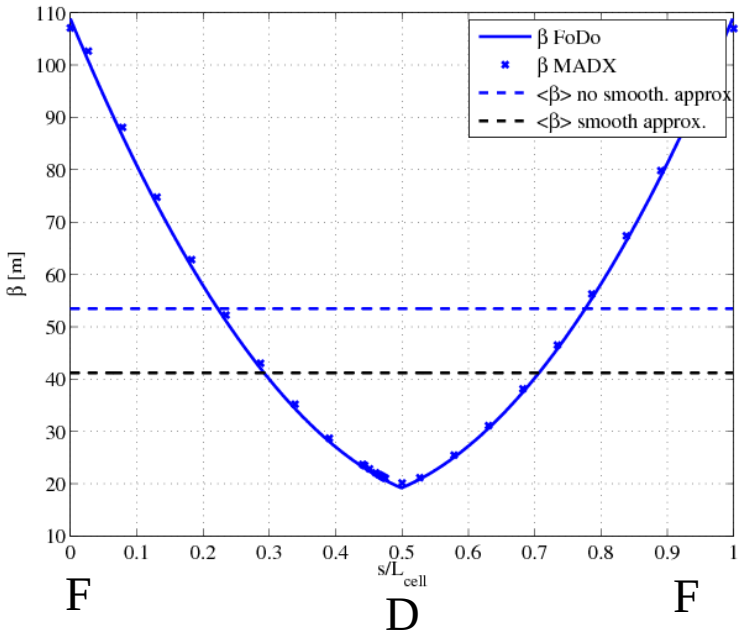
NAME:	ARC
POSITION:	250
INTERACTION:	IMPEDANCE
TYPE:	1
TABLE_TYPE:	4
SCALE:	1
BETX:	$\bar{\beta}$x
BETY:	$\bar{\beta}$y
MULTIPLY:	0
MULTIPLY_COEFF:	1

Means:
"Put the ARC impedance at 250m in the lattice. The beta functions seen by the beam are $\bar{\beta}_x$ and $\bar{\beta}_y$"

- NB: The $\bar{\beta}$ is usually not given by R/Q as in the smooth approximation.

Lattice manager – beta functions

- NB: The $\bar{\beta}$ is usually not given by R/Q as in the smooth approximation.



- In weak focusing machine (where 0<n<1 with n=$\rho^2 K$ with ρ bending radius and K quadrupolar strength), the average beta function could be taken as $\bar{\beta} = R/Q$ with a good approximation (<1%) since the phase advance per cell μ was very small (few degrees).

- In a strong focusing machine (n>1e3) this is no more valid. For the FoDo cell we can calculate:

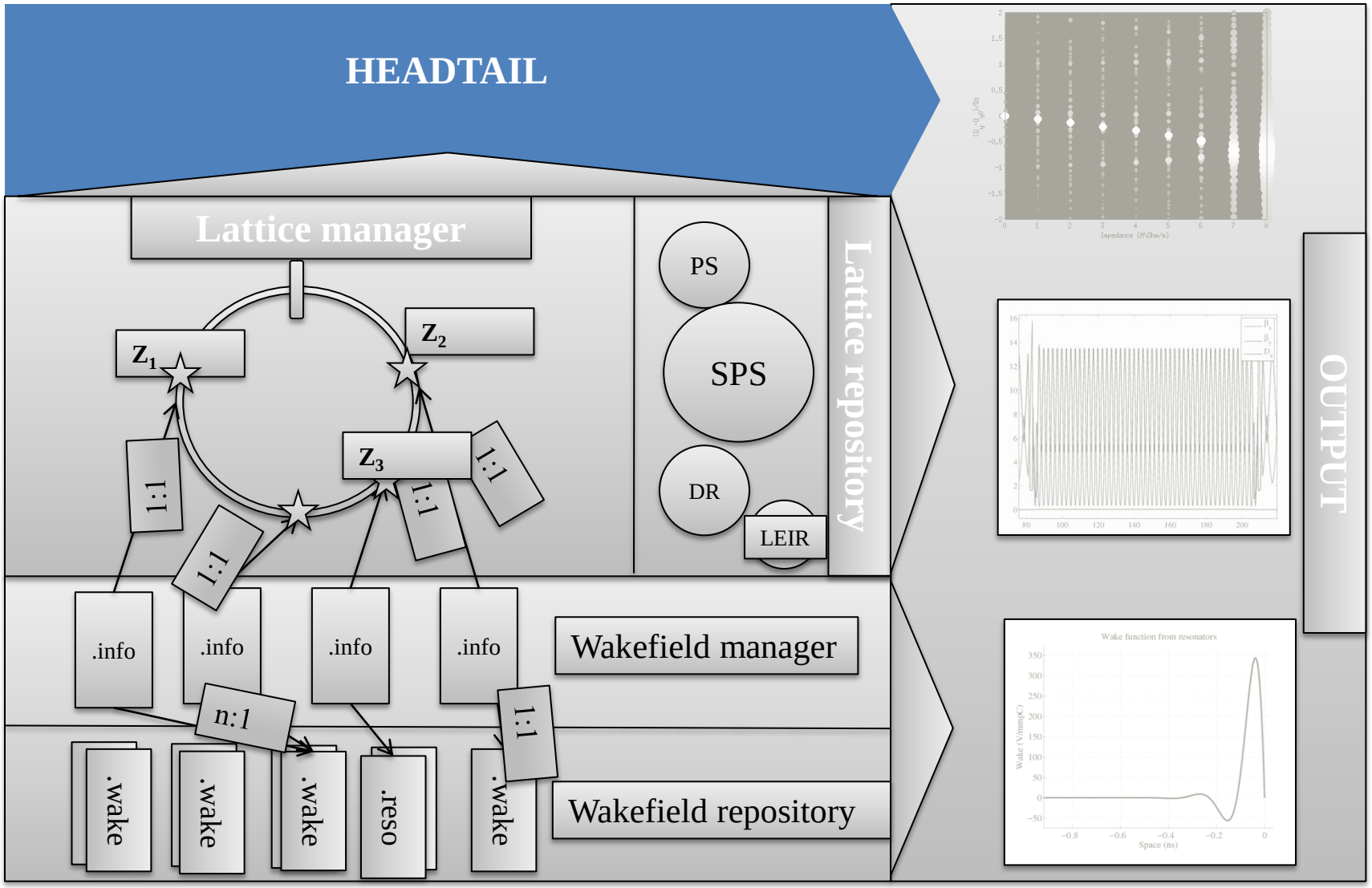
$$\bar{\beta} = L_{cell}\left(\frac{1}{\sin(\mu)} - \frac{1}{6}\tan\left(\frac{\mu}{2}\right)\right)$$

with μ in radiants. In SPS for Q26 from $\bar{\beta}$=41m of the smooth approximation, we get $\bar{\beta}$=52m, i.e. 25% more.

NB If we use a OTM with $\beta_x = \frac{\mathrm{R}}{\mathrm{Q}} = 41m$, the resistive wall impedance of the whole machine will be lumped at the starting point weighted by the ratio $\frac{\bar{\beta}}{R/Q} = \frac{52}{41}$.

$$\Delta Q \propto \oint_0^L W\beta(s)ds = WC\bar{\beta} = \frac{\mathrm{R}}{\mathrm{Q}}\left(\frac{\bar{\beta}}{R/Q}WL\right)$$

HEADTAIL


HEADTAIL
Lattice manager
Z1
Z2
Z3
1:1
1:1
1:1
1:1
PS
SPS
DR
LEIR
Lattice repository
.info
.info
.info
.info
n:1
1:1
Wakefield manager
.wake
.wake
.wake
.reso
.wake
Wakefield repository
OUTPUT
Wake function from resonators
Space (ns)

HEADTAIL - Configuration file

Here the new part of the ".cfg" file.

Specify the folder where your machine can be found.

Put to "**1**" if you want to lump all the lattice at start.
NB: This will work only if all the wakes are tables (".wake"), of the same type, and have the same length.

Flag_for_bunch_particles_(1->protons_2->positrons_3&4->ions):	1
Number_of_particles_per_bunch:	1e11
Machine:	**SPS**
Observation_points:	**BPH+BPV**
Interaction_points:	**MKE+MKP+MKQ+MKD**
Install_impedance:	**REWALL+RFCAVITY+VAC***
Lump_impedance_(1->Yes,0->No):	**0**
Bunch_length_(rms_value)_[m]:	0.2023
Normalized_horizontal_emittance_(rms_value)_[um]:	3.0
Normalized_vertical_emittance_(rms_value)_[um]:	3.0
[….]	

Specify the impedance that are **already present** in the MADX file with their MADX name :

- "MKPA.11931" (only this impedance)
- "MKP*" (all MKP)
- "MK*" (all impedance starting with MK)
- "NONE" (nothing)
- The name should match the corresponding ".info" file (i.e. "MKE.11631.info", etc..)

Specify the BPMS with their MADX pattern:

- "BPV.10010" (only this monito)
- "BPV*" (all monitor starting with BPV)
- "BPV+BPH" (all hor. and vert. monitors)
- "NONE" (nothing)

Specify the impedance that are **not present** in the MADX file and that will be installed:

- The name should match the ".info" file (i.e. REWALL.info, RFCAVITY.info)
- The info file will contain the type of impedance and the place you want to install it.
- NB: you could choose an already "occupied" position. In this case MADX will automatically shift your element to find a free space
- "NONE" (nothing)

Note: All the indirect information about the machine have been removed like:

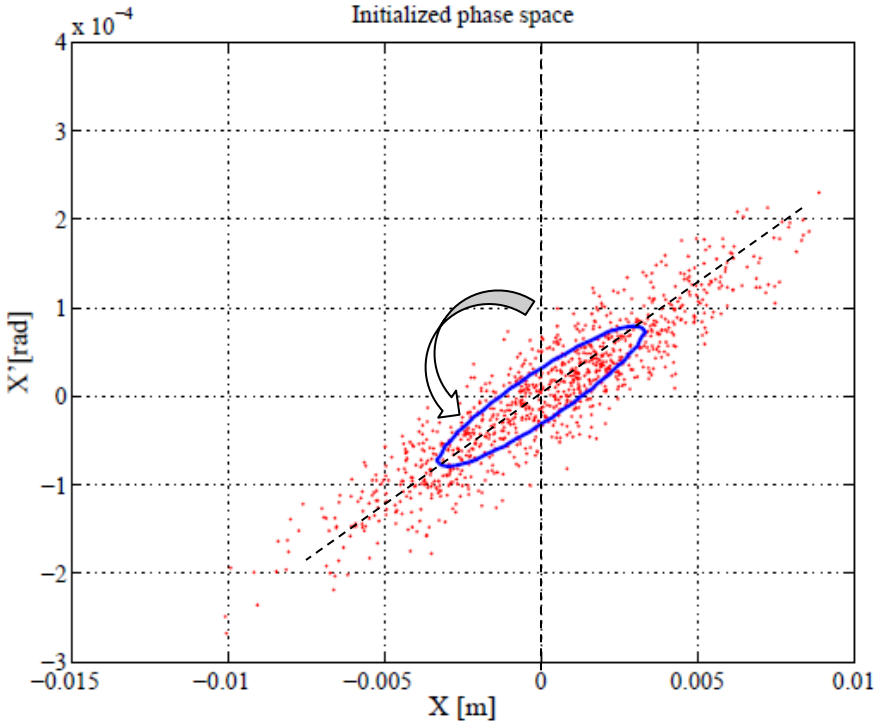
- Circumference (taken from MADX).
- Momentum compaction factor (taken from MADX).
- Average beta X and Y (calculated from MADX).
- Average beam pipe width and height (if present, taken from MADX).

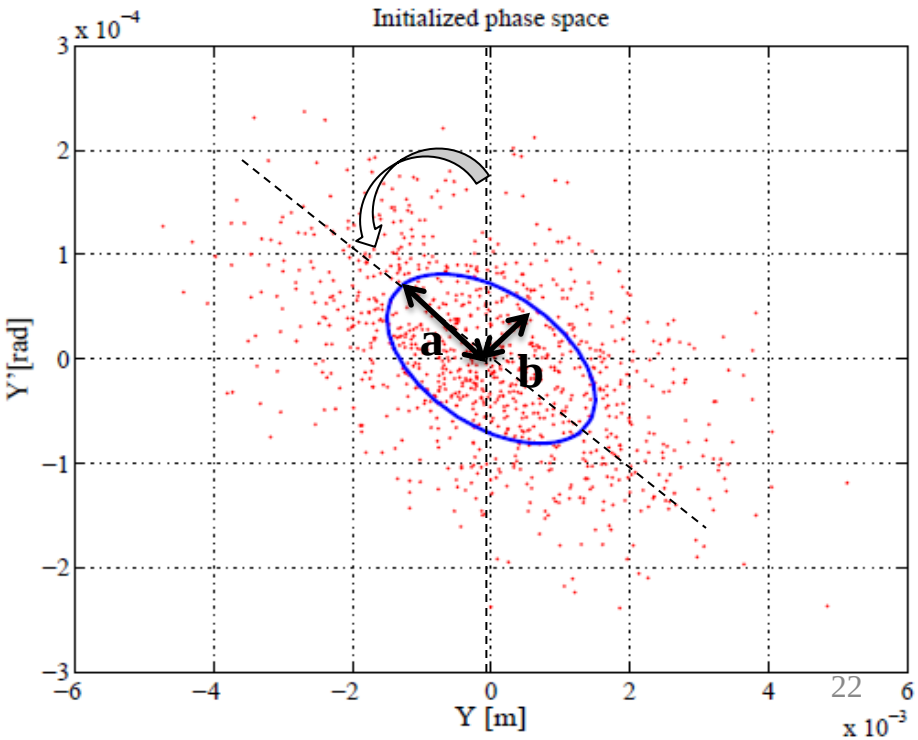
If no observation points are specified then the observation will be at START.

HEADTAIL - Particles ensemble generation

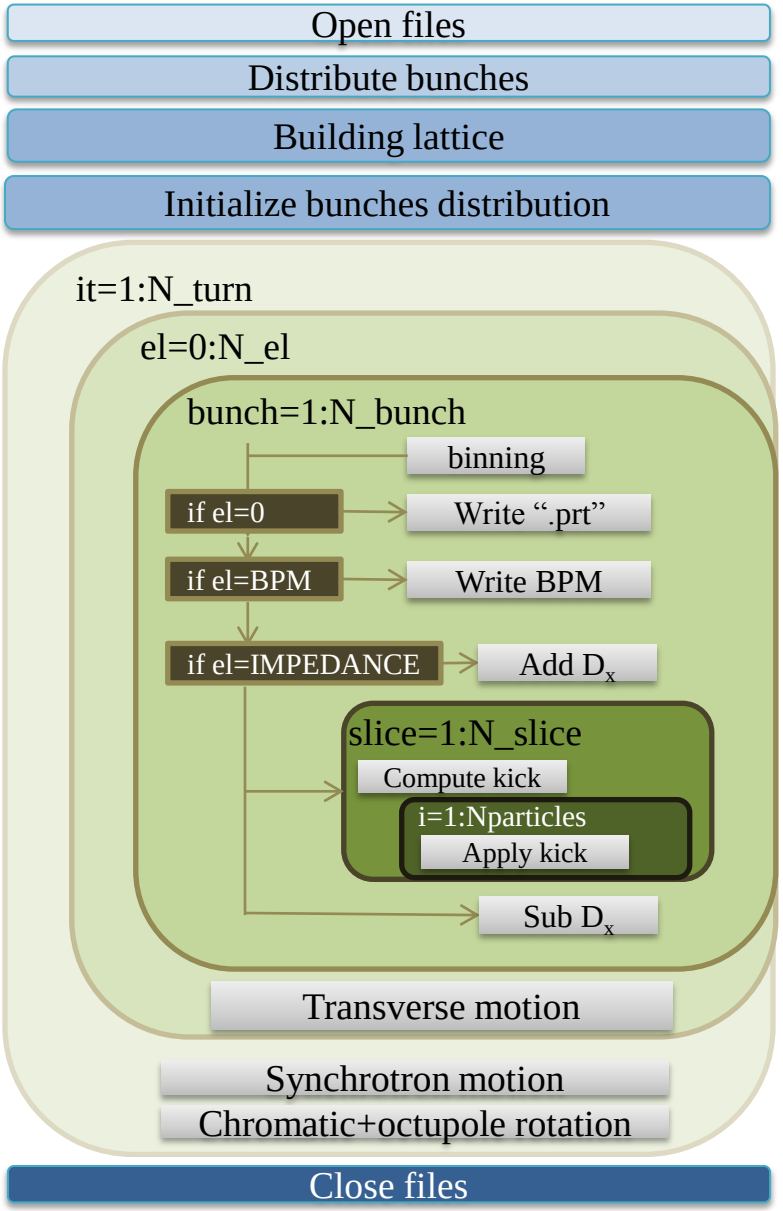
Some modification had to be done in the particle generation. Since START can be a skew point in the lattice:

1. We calculate the **a** and **b** axes of the skew phase space ellipse;
2. We generate the particle ensamble with the Box-Muller transformation to have a gaussian distribution with the specified emittance.
3. We tilt the distribution following the angle obtained using the twiss parameters.

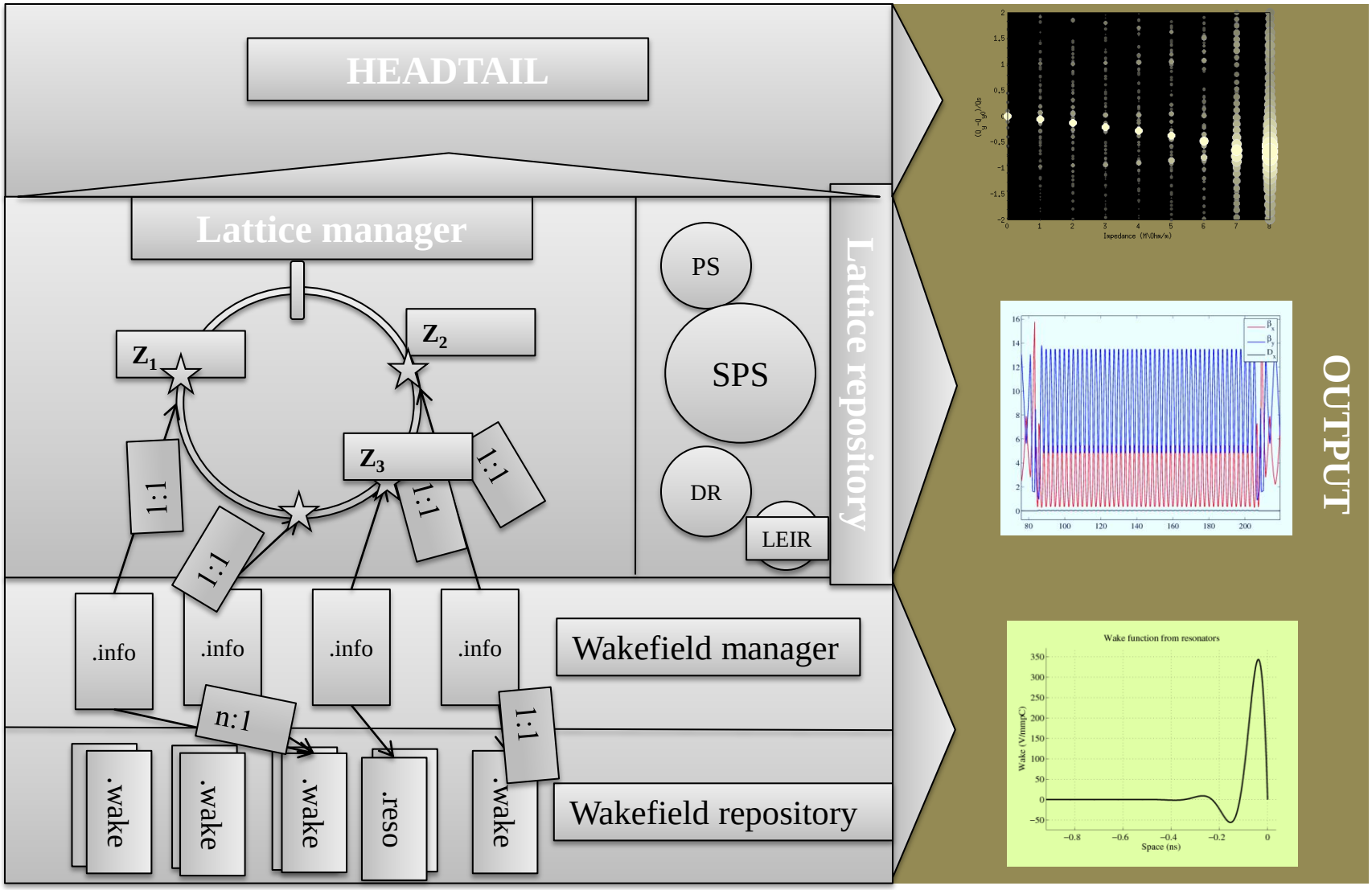




HEADTAIL – MAIN LOOPS



Output


HEADTAIL
Lattice manager
Z1
Z2
Z3
1:1
1:1
1:1
1:1
PS
SPS
DR
LEIR
Lattice repository
.info
.info
.info
.info
n:1
1:1
Wakefield manager
.wake
.wake
.wake
.reso
.wake
Wakefield repository
OUTPUT
Impedance (R\(hw/w)
Wake function from resonators
Wake (V/mm/pC)
Space (ns)

Output

All the informations/results are collected in a specific folder named "Output_#NAME_CFG_FILE".
For example, running test.cfg, all the results will be in Output_test.cfg.

HDTL_average_lattice.dat: MADX twiss file for the complete machine you simulated.
HDTL_selected_lattice.dat: MADX file for the elements were selected for the tracking.

Prb.dat: Takes a snapshot of the phase space (id, x,x',y,y',z,z') at the turns selected in the config file. 100 particles are taken randomly along the first bunch.
Pini.dat: Takes a snapshot of the initialized phase space (x,x',y,y',z,z'), 1000 particles are taken randomly.
Bunchds.dat: Takes a snapshot (s[m], Npr(s))of the longitudinal distribution for the first bunch after all the interaction in the turn have been made at the turns selected in the *".cfg"* file. The bunch extends from -5σ to +5σ (Verteil function).
Prt.dat: General tracking information files. In this version will refer to the START point in the lattice. The columns are:

1) Time_step: sampled time in which the beam passes at START.
2) <X>: average centroid X [m].
3) <Xp>: average centroid X' [rad].
4) <Y>: average centroid Y [m].
5) <Yp>: average centroid Y' [rad].
6) <Z>: average centroid Z [m].
7) <dp/p>: average centroid dp/p [adim].
8) $<\sigma_x>$: average horizontal beam size* [m].
9) $<\sigma_y>$: average vertical beam size* [m].
10) $<\sigma_z>$: average longitudinal beam extension [m].
11) $<\sigma_{dp/p}>$: average beam momentum spread [adim].
12) ε_{xn}: beam normalized horizontal emittance [mm mrad]**.
13) ε_{yn}: beam normalized vertical emittance [mm mrad]**.
14) ε_l: beam longitudinal emittance [eV s]**.
15) J_x: Horizontal action variable [m]***.
16) J_y: Vertical action variable [m].
17) $\varepsilon_l=4(\pi\sigma_t\sigma_{\Delta E})$ beam longitudinal emittance [eV s]****.
18) ρ_{yz} : Y-Z correlation.
19) Effective number of particles (N_tot-N_lost)/N_tot.

*since START can have alfa xy different from zero, these sigma are still not correct since where referred to a flat ellipse.
** calculated as $\varepsilon = \sqrt{\langle x^2\rangle\langle x'^2\rangle - \langle xx'\rangle^2}$ for a centred beam.

*** $J_x = \frac{1}{2\beta}(x^2 + (\beta x' + \alpha x)^2)$.

**** This is an approx. for a beam whose dimension in longitudinal phase space are smaller in comparison to the bucket dimension.

Output

#Name_imp.track: File containing the wake used in the simulated interactions.

- For a **broad band** impedance is made of 3 columns:
  (s[ns], Wtrasv(V/mm pC), Wlong(V/pC)),
- For a **table impedance** is made of coloumns (s[ns], Wx_{dip}(V/mm pC), Wy_{dip}(V/mm pC):
  Wx_{quad}(V/mm pC), Wy_{quad}(V/mm pC), Wxy_{dip}(V/mm pC), Wxy_{quad}(V/mm pC), Wx_{const}(V/pC), Wy_{const}(V/ pC), Wlong(V/pC)),

Beware: there's a sign change in x and y coordinates since HEADTAIL uses this different convention.

Sample.dat: Net bunch energy loss per turn in case of longitudinal impedance. (turn number, dp_turn [MeV], bunch_id).

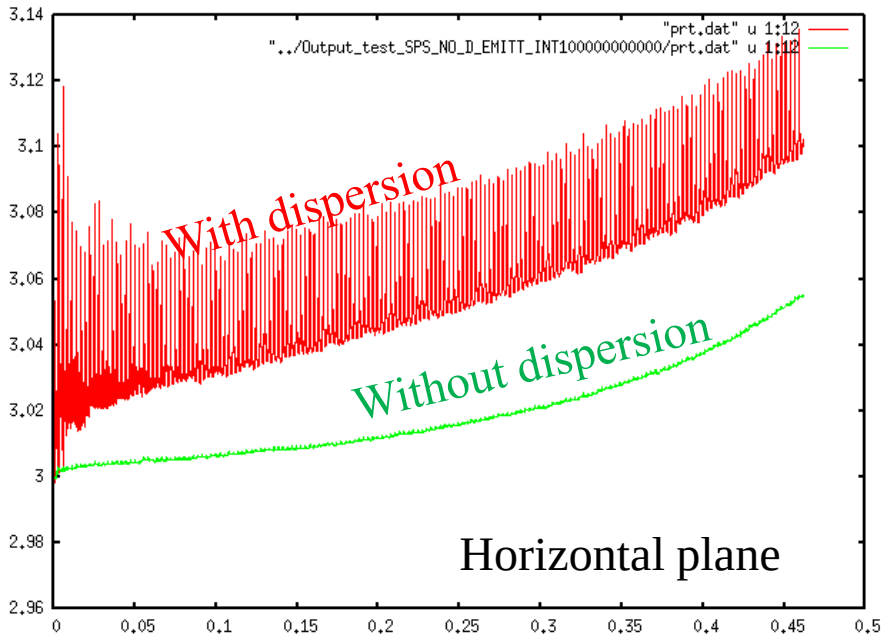
Hdtl.dat: Headtail monitor emulator. It contains a table with these columns:

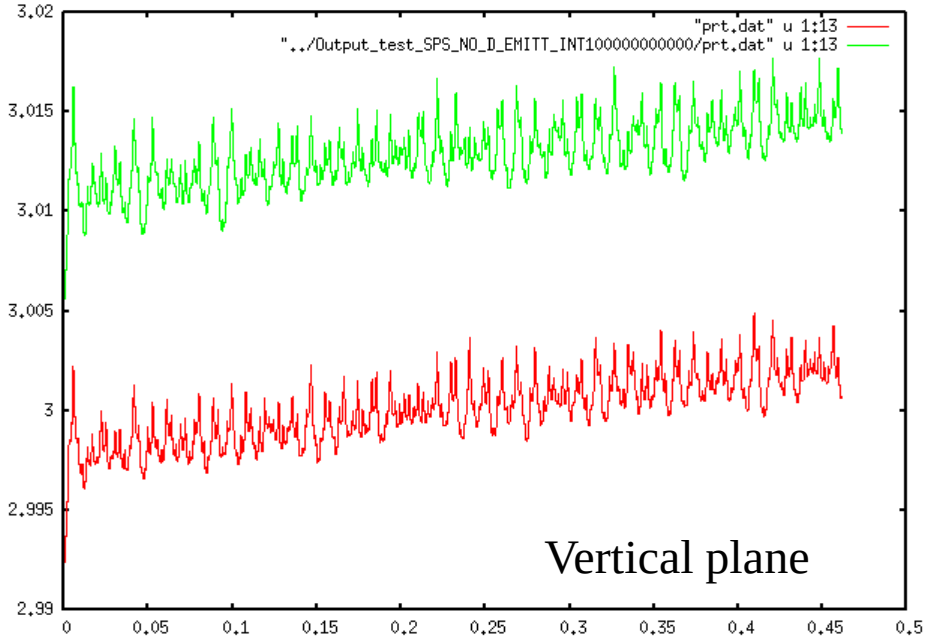
1) Spatial distribution [ns] from -Nσ to Nσ shifted of Nσ (ie. From 0 to 2Nσ) .
2) Npr_{slice} * <X_{slice}>: Number of particle per slice times its displacement in X.
3) Npr_{slice} * <Y_{slice}>: Number of particle per slice times its displacement in Y.
4) Npr_{slice} * < σx_{slice}>: Number of particle per slice times its rms size in X.
5) Npr_{slice} * < σy_{slice}>: Number of particle per slice times its rms size in Y.
6) Npr_{slice} :Number of particle per slice
7) d(Npr_{slice})/dz: Longitudinal derivative of the distribution per slice.
8) Bunch_id number.

Inph.dat: Collects informations about the longitudinal matching number, total number of macroparticles, bunch and slices used in the simulation, the percent of beam loss and the pipe average apertures (rpipex, rpipey).

lumped.wake: if lumping is chosen, in this file will be printed the total wake weighted at START.

Observations on emittance growth



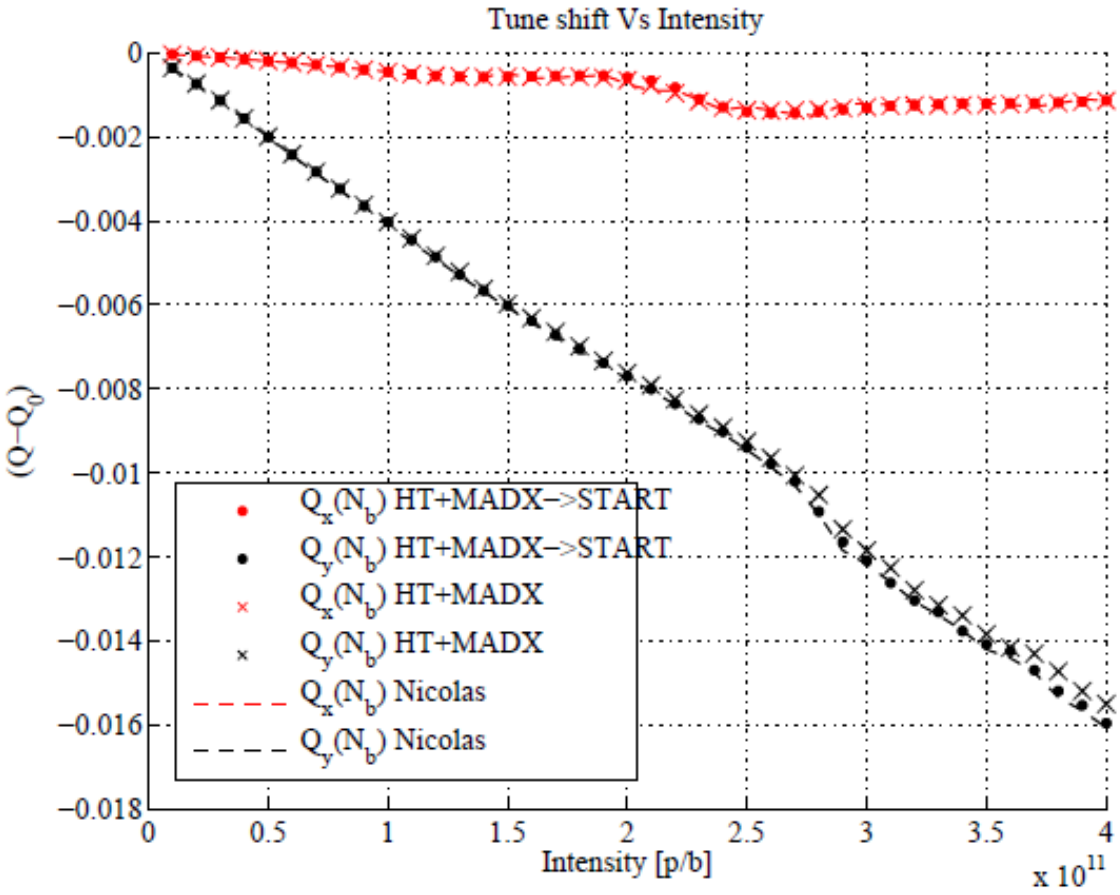


- While in the vertical plane we observe basically no emittance growth, in the horizontal plane it grows of +1.3% in 20000 turns (all kickers impedance, Nb=10^11ppb).
- The curves are shifted because of the initial distribution difference.
- The dispersion is added and then subtracted before the interaction.

Is the wake energy taken out from the beam?

Kickers as benchmark

As a first check, we compared with Nicolas version the tune shift induced by the distributed and lumped kicker impedances calculated by Carlo. The lattice is a SPS-Q26 lattice.



Outlook

- The HEADTAIL code can now support different lattices, as well as lumped or distributed impedances. This version can be found at: https://svnweb.cern.ch/cern/wsvn/hdtldev/branches/HDTLattice/
- The way it interfaces with MADX and build the element structs, allows easy insertions of other kind of elements like ecloud, space charge kicks, etc…
- The wakefield management could to be further improved in order to be interfaced with **Zbase**.
- A list of **benchmarks** and work to do should be now defined and compared with the multi bunch single lattice version:
    1. Simulations of rise-times (Nicolò),
    2. Simulations of tune shifts in transverse plane for DR (Eirini),
    3. Simulations of potential well distortion for PS (Mauro),
    4. Simulations with first and second order chromaticity,
    5. Restore and simulate the multi-bunch part of the code (Nicol[as+ò]),
    6. Simulations for impedance localization (Nicolò),
    7. Including and testing the PSB and LHC lattice,
    8. Including a layer in the wakefield management for interpolating the wakes.
    9. Checking the octupole detuning (include real lattice elements since in SPS they are only 3)
    10. Simulation for emittance growth w/o quadrupolar component, maybe following the evolution of a little volume of phase space.
- If we'll be interested in tracking simulations of non-linear elements and impedances, we should consider to interface HEADTAIL with PTC.
- Create a Test-folder in the HDTLattice branch where people can collect all their benchmark configuration files and results.

Many thanks!

Particular thanks to:
F.Antoniou (DR optics),
H.Bartosik (Optics),
O.Berrig (LEIR lattice),
C.Hernalsteens (PS lattice),
E. Koukovini Platia (DR studies),
K. Li (FFT processing),
M.Migliorati (Usability),
Y. Papaphilippou (DR optics),
S.Persichelli (Usability),
R. Tomás (Optics),
C.Zannini (SPS studies).

Lattice funtions (Lattice.h)

- All the function concerning lattice generation were gathered in a new header file **Lattice.h**. This hands also the wake generation.

Function definitions for different kinds of wake fields	
Average lattice	Function to calculate the average parameter for selected lattice. Momentum compaction factor, circumference, average beta X Y, average apertures.
Build Matrix	Function that construct the transport matrix between two elements using their twiss functions.
Print Matrix	Print lattice matrixes
Prod Matrix	Products between two matrices
Prod Vector	Products matrix vector for transport.
One turn map	Calculate OTM for crosscheck or ilump=1
Write madx	Writes down the MADX file to get information about the observers and impedances specified in the .cfg file.
Build wakes	Associate wakes to specified impedances elements. Lumps at START is ilump=1
Build Lattice	Reads the madx output file and construct the lattice elements
DeltaQ_octuple	Apply the rotation due to octupole detuning
DeltaQ_chromaticty	Apply rotation due to chromaticity, first and second order.

HEADERS in HEADTAIL and their function

Constants.h

Main physics variables (pi, c,e,mu0, e0,mp,me,rp,re,I0,Z0)

Bunch.h

Functions for bunch assembly and re-assembly	
update_bvalues	
binning	sub-divides the bunch into thin slices interact one by one with the kicking wake.

Variables.h

All the global variables used in the code.

HEADERS in HEADTAIL and their function

Fields.h

Function definitions for different kinds of wake fields	
wake_func	Function that gives the transverse wake function at a given location z < 0 (z is 'pos') for a resonator
wake_reswall	Function that gives the transverse wake function at a given location z < 0 (z is 'pos') for a resistive wall (classic thick wall formula - see W. Chao's book"Physics of Collective Beam Instabilities in High Energy Accelerators", p.71)
wake_reswall_ib	Function that gives the transverse wake function at a given location z < 0 (z is 'pos') for a resistive wall with inductive by-pass (see A. Koschik PhD, p.61)
Wake_funcz	Function that gives the longitudinal wake function at a given location z < 0 (z is 'pos') - single bunch in cavity.
locate	Function that gives the position lprov (integer) in table, such that table[lprov-1]<z<table[lprov]
wake_table	Function that gives the wake function at a given location z < 0 (z is 'pos'). Wake coming from table
wake_pretreat	Function that gives zout, a table of distances z such between 2 sucessive z in this table, wake functions do not vary by more than "ratio" (in relative)

HEADERS in HEADTAIL and their function

Function definitions for different kinds of wake fields	
wake_func	Function that gives the transverse wake function at a given location z < 0 (z is 'pos') for a resonator
wake_reswall	Function that gives the transverse wake function at a given location z < 0 (z is 'pos') for a resistive wall (classic thick wall formula - see W. Chao's book"Physics of Collective Beam Instabilities in High Energy Accelerators", p.71)
wake_reswall_ib	Function that gives the transverse wake function at a given location z < 0 (z is 'pos') for a resistive wall with inductive by-pass (see A. Koschik PhD, p.61)
Wake_funcz	Function that gives the longitudinal wake function at a given location z < 0 (z is 'pos') - single bunch in cavity.
locate	Function that gives the position lprov (integer) in table, such that table[lprov-1]<z<table[lprov]
wake_table	Function that gives the wake function at a given location z < 0 (z is 'pos'). Wake coming from table
wake_pretreat	Function that gives zout, a table of distances z such between 2 sucessive z in this table, wake functions do not vary by more than "ratio" (in relative)

HEADERS in HEADTAIL
and their function

Files.h

Helper functions	
Reverse	Reverse string
itoa	Integer to string

File functions	
open_files	Open files for storing calculation data.
close_files	Close output files.
read_data	Define main parameters by reading data from specified configuration-file

HEADERS in HEADTAIL and their function

Initialisation.h

Function definitions used by the main program	
momentum	Function that gives the momentum evolution during acceleration
gammarel	Function that gives the relativistic gamma evolution during acceleration
betarel	Function that gives the relativistic beta evolution during acceleration
rfpot	single rf potential
randDistrVal	random value from elliptical or gaussian distribution (phi, phiDot, ws0, phiDotMax, phiS, distFlag)
etaslip	Function that gives the slip factor eta evolution during acceleration

HEADERS in HEADTAIL and their function

Functions for the generation of cloud and bunch distributions	
verteil	Vector containing the quantity whose distribution we want to plot, number of elements of this vector
verteil2	Calculate arbitrary distribution
parabolic	Parabolic particle and velocity distribution in longitudinal direction.

Functions for the initialisation of the simulation parameters and the cloud and bunch distributions	
init_values	Initialize all parameter values, necessary for calculation.
Initialization()	Initialize particles positions and velocities
refresh_el	Re-initialization of the electrons distribution after one full bunch passage
new_fraction	Re-initialization of the fraction of electrons newly generated after one slice
gsl_qrng_sample	Function for quasi-random sampling of bunch particles using a hammerslay sequence