Upgrade of Makethin

- **TEAPOT slicing algorithm, extended to n > 4 as reported in LCU meeting 18/09/2012, working well, described in contribution currently prepared for IPAC 2013**

Here mainly discussing plans and preliminary tests for three further upgrade steps

- **upgrade in functionality, allowing to turn off slicing by selection statements**
- **thick quadrupole slices**
- **automatic transfer of bending magnet fringe fields to dipedge elements**

also major internal changes (using C++ with standard library strings, vectors)

speed improvements, faster search for elements

internal changes should be transparent in standard use

`option,debug=true; option,verbose=true;` ! essential for development and debugging

Acknowledgment : Thys Risselada, Frank Schmidt

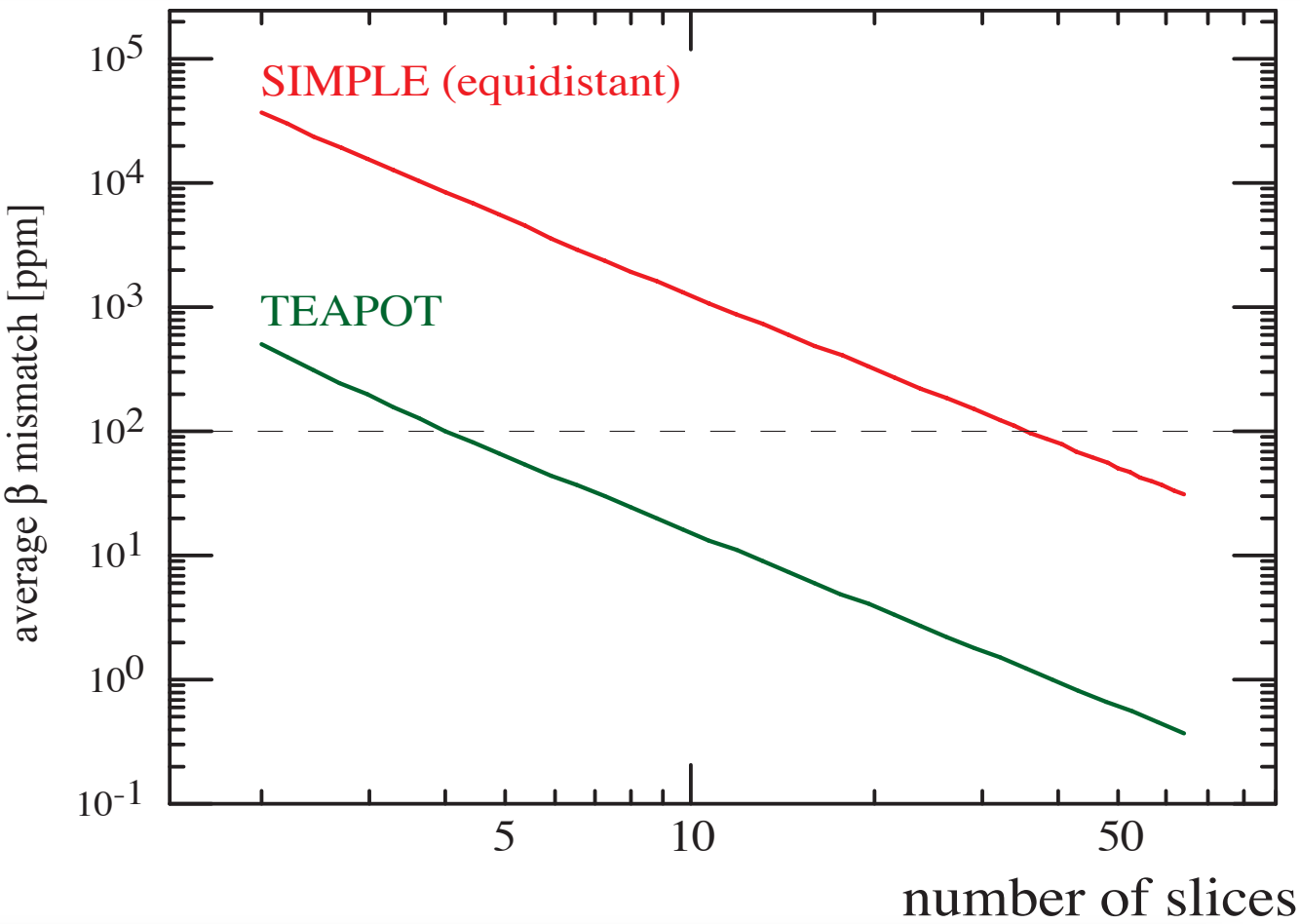
and Riccardo De Maria, Massimo Giovannozzi, John Jowett, Laurent Deniau, Rogelio Tomas, Adriano Garonna

TEAPOT slicing

Note, that for backward compatibility, default slicing goes to simple (equidistant) for n > 4.

Advice is to always use TEAPOT slicing, details see LCU 18/09/2012 and IPAC'13

```
makethin, sequence=lhcb1, style=teapot;
```



plot from Thys Risselada for LHC MQ main quads

For same #slices, **TEAPOT** reduces the β - mismatch by **2 orders of magnitude** for the LHC; 10 × more slices required to get to similar precision with SIMPLE

Upgrade in functionality

```
select,flag=makethin,RANGE=range,CLASS=class,PATTERN=pattern[,FULL][,CLEAR],SLICE=#slices,thick=true;
```

default is slice=1, thick=false

Turn off slicing by selection with a slice number < 1, to just copy the thick element

```
select, flag=makethin, class=sextupole, slice=0; ! turn off slicing of all sextupoles
select, flag=makethin, pattern=mbxw\., slice=0;  ! keep mbxw dipoles thick
```

Thick slicing option, effective for quadrupoles and bends (ignored for others)

```
select, flag=makethin, class=quadrupole, thick=true , slice=2; ! thick quadrupole slices
select, flag=makethin, class= rbend, slice=1, thick=true; ! translates rbend to sbend +
                       dipedge
```

for bends for the moment restricted to 1 thick slice (is there demand for more ?)

example, LHC triplet thick slicing

```
select, flag=makethin, pattern=mqxa\., slice=2, thick=true;
select, flag=makethin, pattern=mqxb\., slice=4, thick=true;
```

Selection works on the current sequence (from last **use, sequence =** *seqname* **;**)
The information thick = true or false and the number of slices selected for makethin is stored internally with each element

Thick quadrupole slicing

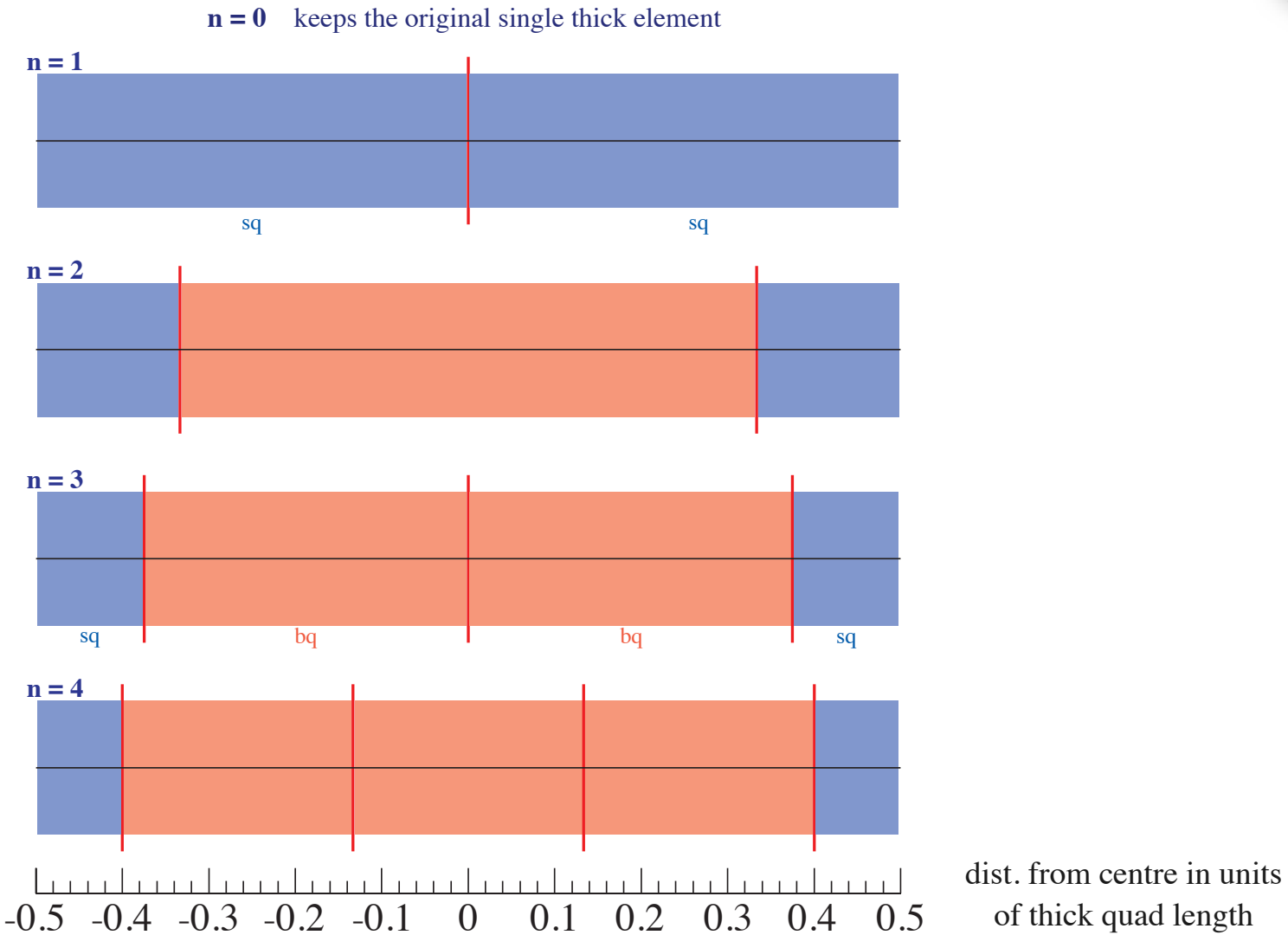


Illustration of thick quadrupole slicing, (using TEAPOT style).
For n-slices we have n-1 bq pieces, and always one start and end piece sq.
results same as original quadrupole
now possible to add multipoles / errors at transitions

Thick quadrupole slices

Tracking possible with thick bends & quads (in principle)

Thin multipole-kicks required for higher orders

#slice < 1 just echoes the thick quadrupole

Now possible to produce n-thick quadrupole slices, can then insert multipole pieces as for thin

Example LHC

```
mqxa: quadrupole, l:=l.mqxa;
mqxa.1r1: mqxa, at = 26.150000, from = ip1,polarity:= 1. ,k1:=kqx.r1 + ktqx1.r1;
```

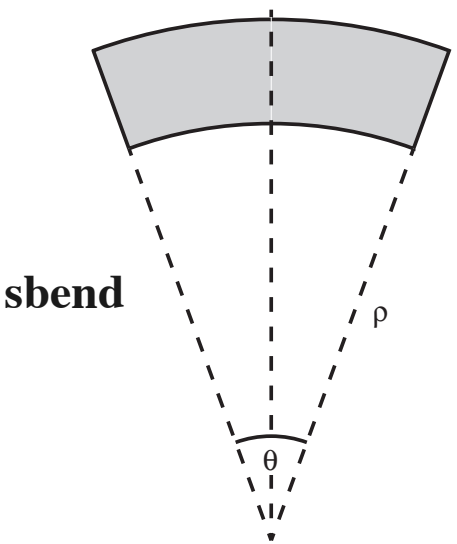
makethin with select, flag=makethin, pattern=mqxa\., slice=2, thick=true; Result (without slot,kmax.)

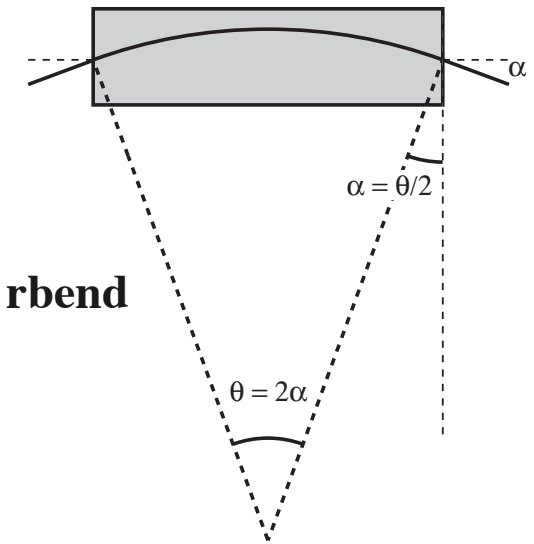
```
mqxa.1r1.sq: quadrupole,l:=( l.mqxa ) * ( 0.16666666666667 ) ,polarity:= 1,k1:=kqx.r1 + ktqx1.r1;
mqxa..1: multipole,lrad:=( l.mqxa ) / ( 2 );
mqxa.1r1..1: mqxa..1,lrad:=( l.mqxa ) / ( 2 ) ,polarity:= 1;
mqxa.1r1.bq: quadrupole,l:=( l.mqxa ) * ( 0.66666666666667 ) ,polarity:= 1,k1:=kqx.r1 + ktqx1.r1;
mqxa..2: multipole,lrad:=( l.mqxa ) / ( 2 ) ;
mqxa.1r1..2: mqxa..2,lrad:=( l.mqxa ) / ( 2 ) ,polarity:= 1,;
```

placed at

```
mqxa.1r1.sq, at = ( 26.15 ) + ( ( l.mqxa ) * ( 0 - 0.41666666666667 ) ) , from = ip1;
mqxa.1r1..1, at = ( 26.15 ) + ( ( l.mqxa ) * ( 0 - 0.33333333333333 ) ) , from = ip1;
mqxa.1r1.bq, at = ( 26.15 ) + ( ( l.mqxa ) * ( 0  ) ) , from = ip1;
mqxa.1r1..2, at = ( 26.15 ) + ( ( l.mqxa ) * ( 0.33333333333333 ) ) , from = ip1;
mqxa.1r1.sq, at = ( 26.15 ) + ( ( l.mqxa ) * ( 0.41666666666667 ) ) , from = ip1;
```

sbend, rbend, dipedge





more generally
bend with extra entry exit angle

rbend special case
where the extra angle
is half of the bending angle

$$\frac{L_{\text{rbend}}}{L_{\text{sbend}}} = \frac{2\sin\frac{\theta}{2}}{\theta} = 1 \quad \frac{\theta^2}{24} + \frac{\theta^4}{1920} \quad ..$$

general bend with fringe fields =

left dipedge × sbend without fringe fields × right dipedge

Comment on practical challenges in changing MAD-X code

Example : bend to dipedge-sbend-dipedge conversion. Idea is

move any nonzero e1 from bend to e1 of new left dipedge element

move any nonzero e2 from bend to e1 of new right dipedge element

for rbend create always dipedges left/right and add angle/2 to e1's of dipedges

change rbend keyword to sbend

in present MAD-X this is very complicated

Many levels of c-structures with pointers to c-structures and use of global variables

(rather than well defined objects)

Attributes can be pointers to expressions w/o value or value, if not given taken from parent .. base_type

Elements defined in several steps and re-used in many places : Work with "clones" and keep

the original information. At the end add location information to elements and place them as nodes in the sliced sequence.

No sanity check / garbage collection. Easy to end up in situation where twiss of the sliced sequence after makethin gets one result and reading back the sliced sequence another result.

Deal automatically with special options

```
rbarc (default=true), on input L = straight length, what is needed for optics, was default in MAD8 and can be
given as input with rbarc=false is the curved length which is longer by  angle / (2 sin ( angle /2 ))
translate any straight length value or expression to the curved length, at (central position) same
start/end position (end normally used in twiss) modified --   can be very confusing

Code now  mad_mkthin.cpp,   internal global stuctures thin_lookup, thin_sequ_lookup removed,
required information now part of objects (C++ classes) within  makethin
grown from 1500 to 2300 lines densely written code, of order 20% for debug printing and statistics
```

Automatic transfer of bending magnet fringe field to dipedge

As described in the makethin documentation dipedge should be used to take care of the edge fields of dipoles which are otherwise lost in slicing.

Proposal for upgraded makethin : by default split off automatically any edge focusing to new dipedge. No dipedge for sbends generated in case of e1=0).

Example, from original thick

```
mb1: sbend, L:=l ,Angle:=ang ,K1:= k1 , E1:= e1 , E2:= e2, hgap := gap, fint:= fin ;
```

to sliced version after makethin

```
mb1.edge_l: dipedge, h:= ang/l, E1:= e1, hgap := gap, fint:= fin ; ! new dipedge at start
mb1: sbend,l:=l , Angle:=ang , K1:= k1 ;                          ! bend with edge effects removed
mb1.edge_r: dipedge, h:= ang/l, E1:= e2, hgap := gap, fint:= fin ;  ! new dipedge at end
```

after thin slicing, the bend becomes

```
mb1: multipole,lrad:=l ,knl:={ ang , k1 * l };
```

In case of rbend, automatically adds angle/2 to e1 of dipedge

Other non-default settings present in bend are transferred to dipedge :

polarity, tilt, mech_sep, v_pos, magnet, model, method, exact, nst

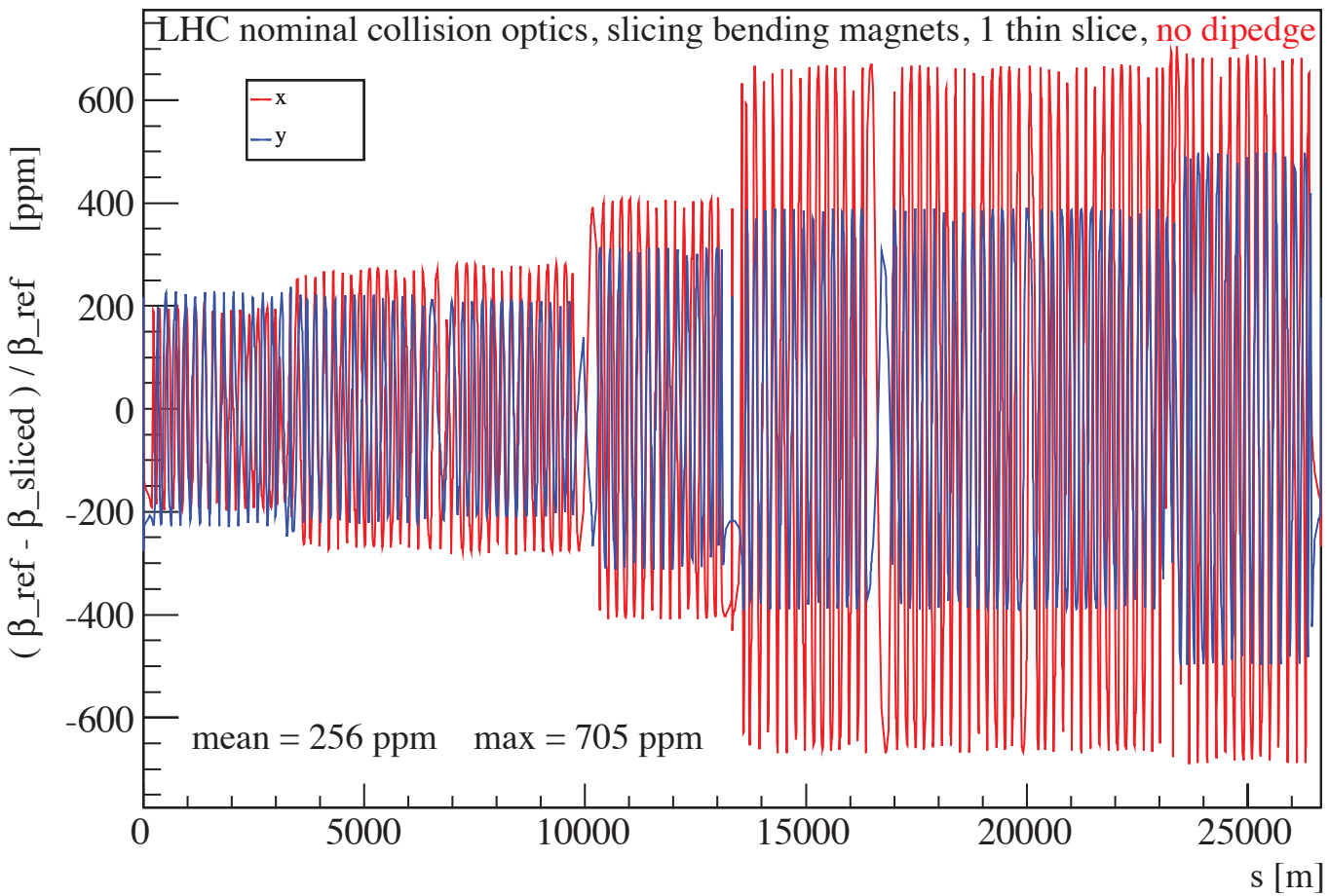
slicing option, examples

```
select, flag=makethin, rbend, thick=true; ! keep translated rbend thick
```

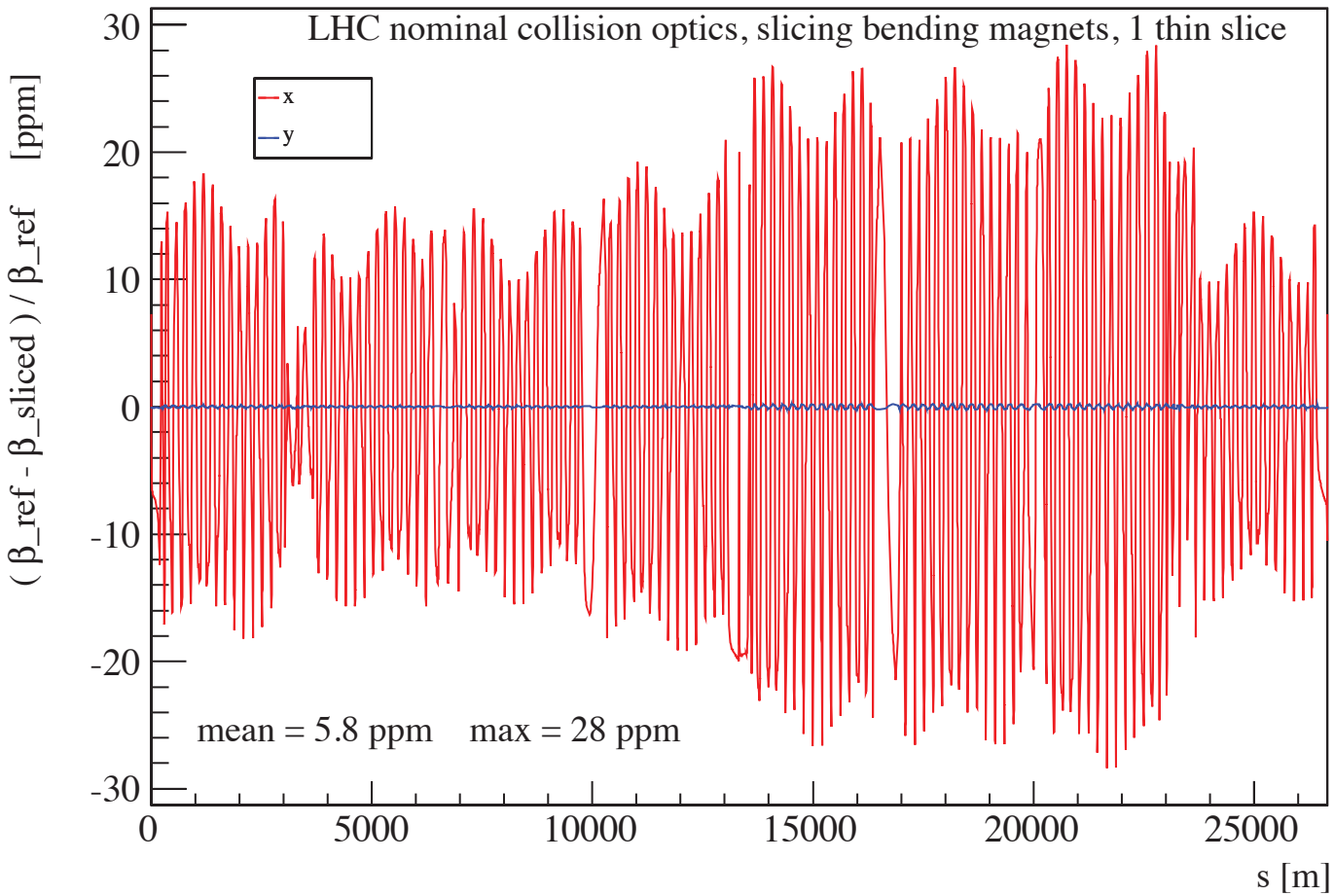
or,

```
select, flag=makethin, rbend, slice=4; ! 4 multipoles slices (with dipedge)
```

Single thin bend slices, LHC, no dipedge



Single thin bend slices, LHC, with dipedge



β - mismatch, completely negligible in y (no bending in y)

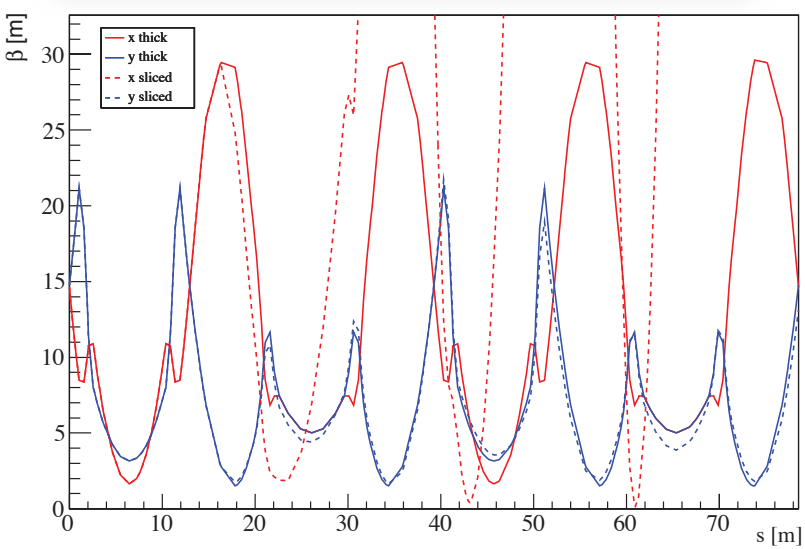
Reduced by factor of 20 in x, and converging with #slices

Essential for slicing in small machines - example LEIR --->

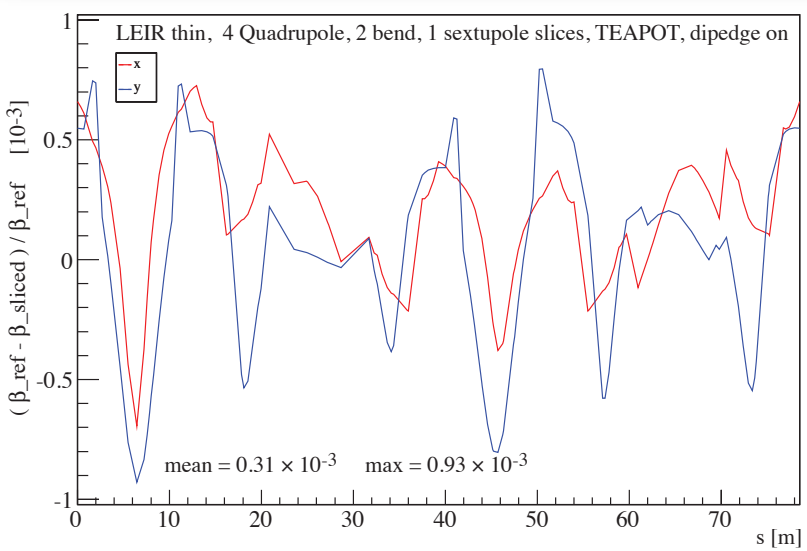
LEIR

Optics files received from
Adriano Garonna on 13/11/2012
No RBEND, no OCTUPOLES
L = 78.544 m
Qx = 1.667, Qy = 2.720

Without dipedges twiss fails,
β's here by twiss with initial values
ΔQ1 = 0.229, ΔQ2 = - 0.033



With dipedge generation on, tunes restored
4 quadrupole and 2 bend slices
get the mismatch below 10^{-3}



increases to mean 3.8%, peak 16 % with 1 thin bend slice

Concluding remarks

TEAPOT slicing works very well, for n slices, the mismatch decreases as $\sim 1/n^2$
The gain for quadrupoles, which are (by far) the main source of mismatch is very significant
--- described in contribution to IPAC'13

when selected, **TEAPOT** is applied to all elements - convergence there as equidistant
Sextupoles, Octupoles - slicing anyway no issue, mismatch from single slice $< 10^{-3}$

Request / proposal to further enhance makethin (in α-testing†) :

- echoing selected elements as thick when #slice < 1
- optionally thick quadrupole slice generation
- automatic dipedge generation for bends, and rbend to sbend generation
  improves (the already sufficiently small) mismatch from bend slicing in the LHC
  essential for small machines

† instructions for testing, on lxplus:

make sure you have access to gcc4.6 libraries

```
export GCCVERSION=4.6 ; export SPI_PLATF=x86_64-slc5-gcc46-opt ; source /afs/cern.ch/sw/lcg/contrib/gcc/4.6/$SPI_PLATF/setup.sh        bash
setenv GCCVERSION 4.6 ; setenv SPI_PLATF x86_64-slc5-gcc46-opt ; source /afs/cern.ch/sw/lcg/contrib/gcc/4.6/$SPI_PLATF/setup.csh        tcsh
```

now run madx64 from my public

```
~hbu/public/madx64
```

Backup

Matrices

$$M_{\text{edge}}(\rho,\alpha)=\begin{pmatrix}1 & 0 & 0 & 0 & 0 & 0\\ \frac{\tan\alpha}{\rho} & 1 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & 0 & 0 & 0\\ 0 & 0 & -\frac{\tan\alpha}{\rho} & 1 & 0 & 0\\ 0 & 0 & 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 0 & 0 & 1\end{pmatrix}$$

Dipedge

$$M_{\text{SBEND}}(\rho,\theta)=\begin{pmatrix}\cos\theta & \rho\sin\theta & 0 & 0 & 0 & \rho\left(1-\cos\theta\right)\\ -\frac{\sin\theta}{\rho} & \cos\theta & 0 & 0 & 0 & \sin\theta\\ 0 & 0 & 1 & \theta\rho & 0 & 0\\ 0 & 0 & 0 & 1 & 0 & 0\\ -\sin\theta & -\rho\left(1-\cos\theta\right) & 0 & 0 & 1 & -\rho\left(\theta-\sin\theta\right)\\ 0 & 0 & 0 & 0 & 0 & 1\end{pmatrix}$$

Thick SBEND no fringe

$$\mathbf{M}_{\text{quad}}(K,L)=\begin{pmatrix}\cos KL & \frac{\sin KL}{K} & 0 & 0 & 0 & 0\\ -K\sin KL & \cos KL & 0 & 0 & 0 & 0\\ 0 & 0 & \cosh KL & \frac{\sinh KL}{K} & 0 & 0\\ 0 & 0 & -K\sinh KL & \cosh KL & 0 & 0\\ 0 & 0 & 0 & 0 & 1 & \frac{L}{\gamma^2}\\ 0 & 0 & 0 & 0 & 0 & 1\end{pmatrix}$$

Thick Quadrupole

All symplectic

using simple notation (agrees with Mad-X for β=1)