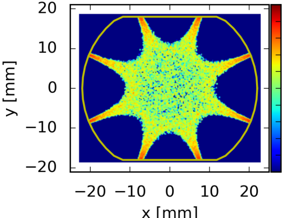
Higher order and skew magnets in PyECLOUD

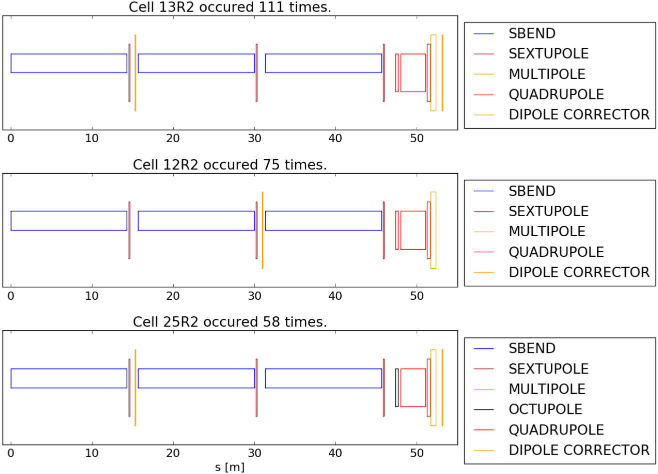
Philipp Dijkstal

May 10, 2017



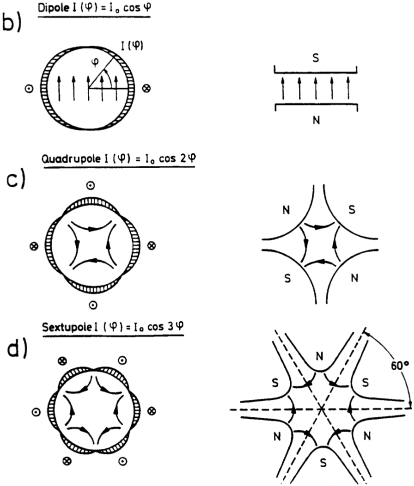
Thanks to G. Iadarola and G. Skripka

Higher order magnets at the LHC



- in every cryogenic half-cell, sextupoles and sometimes octupoles are used
- previously, only drifts, dipoles and quadrupoles were supported

Higher order magnets



$$B_y + iB_x = \sum_{n=0}^{\infty} \frac{1}{n!} \left(b_n + ib_n'\right) \left(x + iy\right)^n$$

- n: order of multipole
- here: dipoles are order 0
- $b_n = \left.\frac{\partial^n B_y}{\partial x^n}\right|_{x,y=0,0}$
- $b_n' = \left.\frac{\partial^n B_x}{\partial x^n}\right|_{x,y=0,0}$
- normal and skew PyECLOUD input parameters

similar to J. Rossbach, P. Schmüser, *Basic course on accelerator optics*, 1993

Implementation

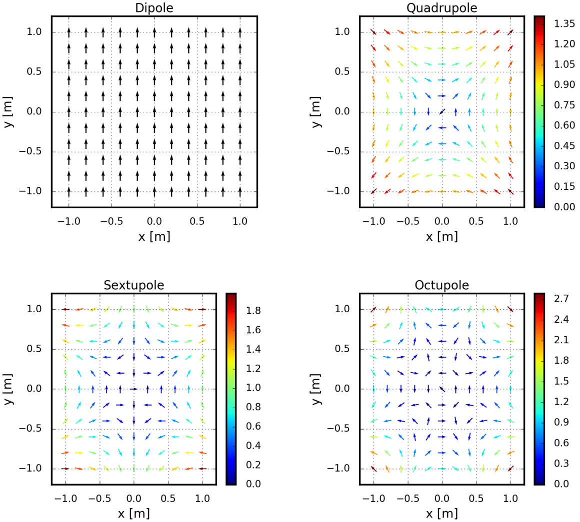
- calculation of B for one particle
- $B_y + iB_x = \sum_{n=0}^{\infty} \frac{1}{n!} \left(b_n + ib'_n\right) (x + iy)^n$
- $n!$ applied during initialization
- $(x + iy)^n$ is built incrementally
- *complex* C datatype turned out to be slow $\rightarrow$ explicit complex calculation for $(x + iy)^n$

Implementation

- calculation of B for one particle
- $B_y + iB_x = \sum_{n=0}^{\infty} \frac{1}{n!} \left(b_n + ib'_n\right) \left(x + iy\right)^n$
- $n!$ applied during initialization
- $\left(x + iy\right)^n$ is built incrementally
- *complex* C datatype turned out to be slow → explicit complex calculation for $\left(x + iy\right)^n$

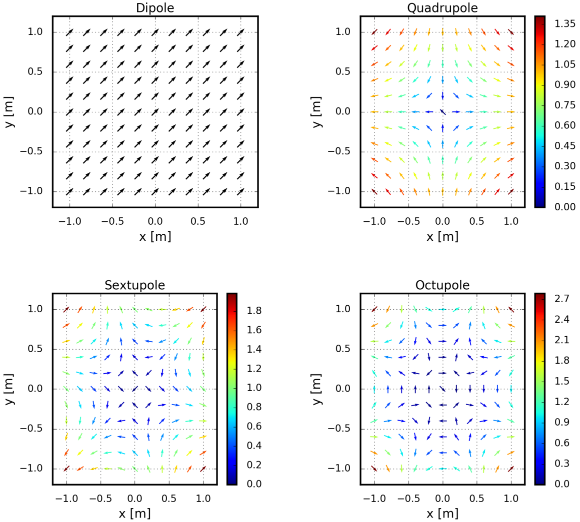
```
rexy = 1.;
imxy = 0.;
By_n = B_field[0]; // Dipole covered here
Bx_n = B_skew[0];
for(order = 1; order < N_multipoles; order++){
    /* rexy, imxy correspond to real,
        imaginary part of (x+iy)^n */
    rexy_0 = rexy;
    rexy = rexy_0*xn1p - imxy*yn1p;
    imxy = imxy*xn1p + rexy_0*yn1p;
    /*
    * Bx +iBy = sum[ (k + ik')(x + iy)^n ]
    * where k, k' are the strengths and skew strengths of the magnet
    */
    By_n += (B_field[order]*rexy - B_skew[order]*imxy);
    Bx_n += (B_field[order]*imxy + B_skew[order]*rexy);
}
```

Field maps



- test boris tracker
- *test_multipole.py* in PyECLOUD code
- equally spaced particles with $\mathbf{v} = v_z$
- $\mathbf{F} = m \cdot \mathbf{a} = e \cdot \mathbf{v} \times \mathbf{B}$
- solve for **B** after one time step

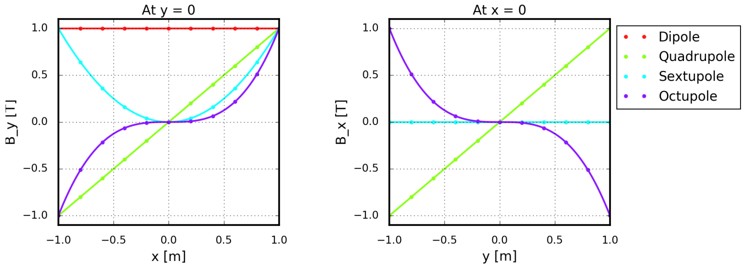
Field maps after rotation



$$B^{(n)}_{\text{multip}}(\varphi) = B^{(n)}_{\text{multip}}(0) \cdot \cos((n+1)\varphi)$$
$$B^{(n)}_{\text{skew}}(\varphi) = B^{(n)}_{\text{multip}}(0) \cdot \sin((n+1)\varphi)$$

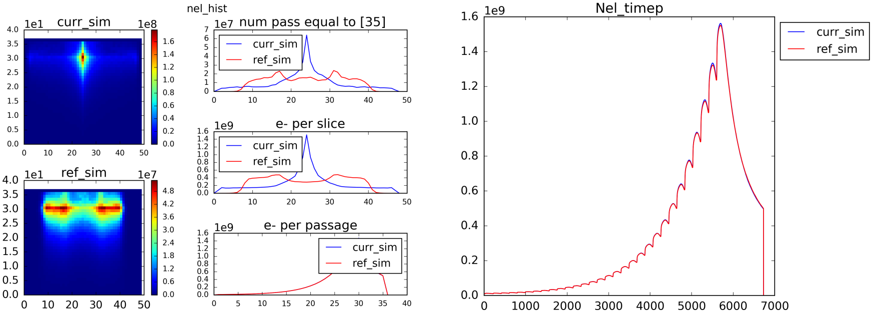
- angle 45°
- consistent field maps

Field strength along axis



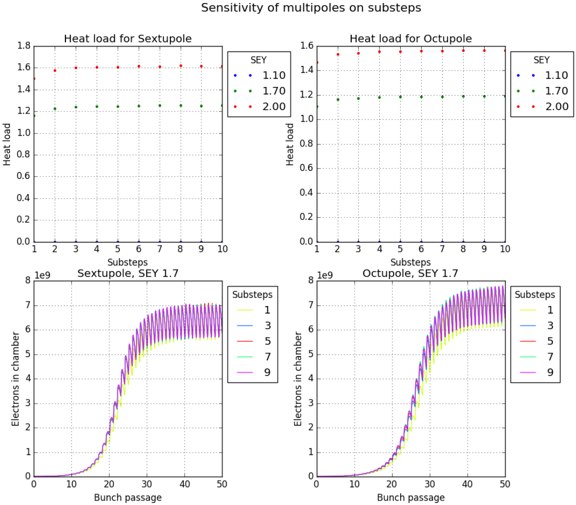
- $B_y^{(n)}(x = 1\ m) = 1$
- no skew component
- matches expected behavior

Comparison of normal vs skew quad in circular chamber



- reference simulation: circular chamber with original field computation
- current simulation: circular chamber with 45° skew quad and new field computation
- test case in PyECLOUD code
- symmetric case, identical outcome expected
- heat loads agree to 0.7%

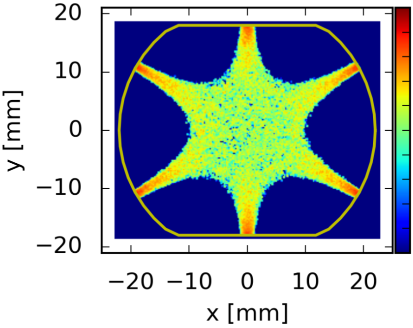
Scan of substeps in time

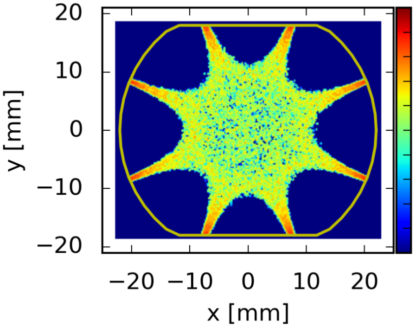


- Do the simulations converge?
- Substeps of MP motion for $Dt = 10^{-11}$ s
- 5 is a good choice

Movies

- 2 gif files on indico
- electron densities in octupole and sextupole
- one bunch passage





Usage

- from PyECLOUD documentation

B_multip	Magnetic momenta. For example: to simulate a dipole with B=8.3 T B_multip = [8.3]; to simulate a quadrupole with gradient 12 T/m set B_multip = [0., 12.].
B_skew	The skew parameters can be included through B_skew. For a skew quadrupole, set B_multip to [0.,0.] and B_skew to [0.,12.].

- example of sextupole:

```
B_multip = [0,0,6.31e4]
B_skew = None
```

Usage - conversion from normalized field strengths

- convert k_i (e.g. from MADX) to B_{multip}
- same procedure for k_{skew} to B_{skew}
- k: magnetic field derivative on x axis, normalized by beam rigidity $B\rho$
- for order n:

$$\underbrace{k_n}_{\text{in MADX files}} = \frac{1}{(B\rho)} \frac{\partial^n B_y}{\partial x^n}$$

$$\underbrace{\frac{\partial^n B_y}{\partial x^n}}_{\text{PyECLOUD input}} = k_n (B\rho)$$

- from MADX documentation:

K2 The normal sextupole coefficient $K_2 = \frac{1}{B\rho}(\partial^2 B_y / \partial x^2)$. (default: $0\ \text{m}^{-3}$).

K2S The skew sextupole coefficient $K_2 S = \frac{1}{B\rho}(\partial^2 B_x / \partial x^2)$ where (x,y) is now a coordinate system rotated by $-30°$ around s with respect to the normal one. (default: $0\ \text{m}^{-3}$). A positive skew sextupole strength implies *defocussing* (!) irrespective of the charge of the particles, in the (x,s) plane rotated by 30° around s (particles in this plane have $x > 0$, $y > 0$).

Usage - conversion from tilting angle

- tilting angle φ

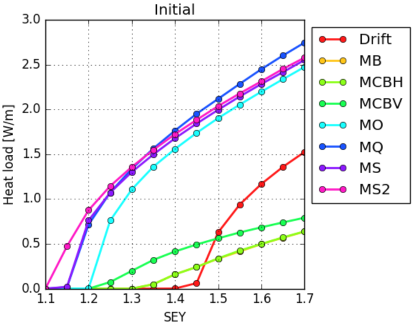
$$
\begin{aligned}
B_{\text{multip}}(\varphi) &= B_{\text{multip}}(0) \cdot \cos((n+1)\varphi) \\
B_{\text{skew}}(\varphi) &= B_{\text{multip}}(0) \cdot \sin((n+1)\varphi)
\end{aligned}
$$

Higher order and corrector magnets at LHC

- obtain k_2 (sextupoles), k_3 (octupoles) from MADX files or LHC control system

Magnet	Identifier	Variable	Origin	Value
Kicker-V	MCBV	$B_{\text{skew}}(7\ \text{TeV})$	MADX	2.5 T
Kicker-H	MCBH	$B(7\ \text{TeV})$	MADX	2.93 T
Sextupole	MS	k_2	LHC data base	-0.12
Sextupole	MS2	k_2	LHC data base	0.07
Octupole	MO	k_3	LHC data base	1 6

First results - heat load against SEY



- initial seeding
- 6500 GeV
- also show drifts, dipoles and quadrupoles

Summary and future studies

- done: higher order, compound and skew magnets part of PyECLOUD
- validation and documentation complete, version 6.2 to be released today
- different interface in version 6.1, so please use 6.2! (no difference for dipoles and quadrupoles)
- up next: photoemission seeding
- systematic modelling of cells
  $\hookrightarrow$ what are the expected heat loads including multipoles and photoemission seeding?