Sextupolar Magnets in CLIC

Rebecca Glaudell
CERN REU 2013
BE-ABP-CC3

with thanks to:
Rogelio Tomas Garcia
Javier Barranco Garcia

rebecca@glaudell.com

August 8, 2013

Sextupoles in the FFS can be adjusted to minimize χ^2.

Sextupoles have: 9 adjustable strengths
5 adjustable positions

Measure minimization of beam spread with weighted χ^2:

$$\chi^2 = \left(\frac{\sigma_x - \sigma_{x_0}}{40}\right)^2 + \left(\sigma_y - \sigma_{y_0}\right)^2$$

MAPCLASS2 code aides in the optimization of the CLIC lattice.

Allows the user to manipulate the beamline or its transfer map.

MAPCLASS2 can use a beamline generated by MAD-X PTC or generate the map directly.

Methodical **A**ccelerator **D**esign-**X**
Polymorphic **T**racking **C**ode

Runs in Python 2.6, but uses some C++ for speed.

Nelder-Mead Simplex Method can be used to optimize the beamline.

1. Choose independent variables:

   sextupoles strength position

2. Write MAPCLASS2 function for dependent variable:

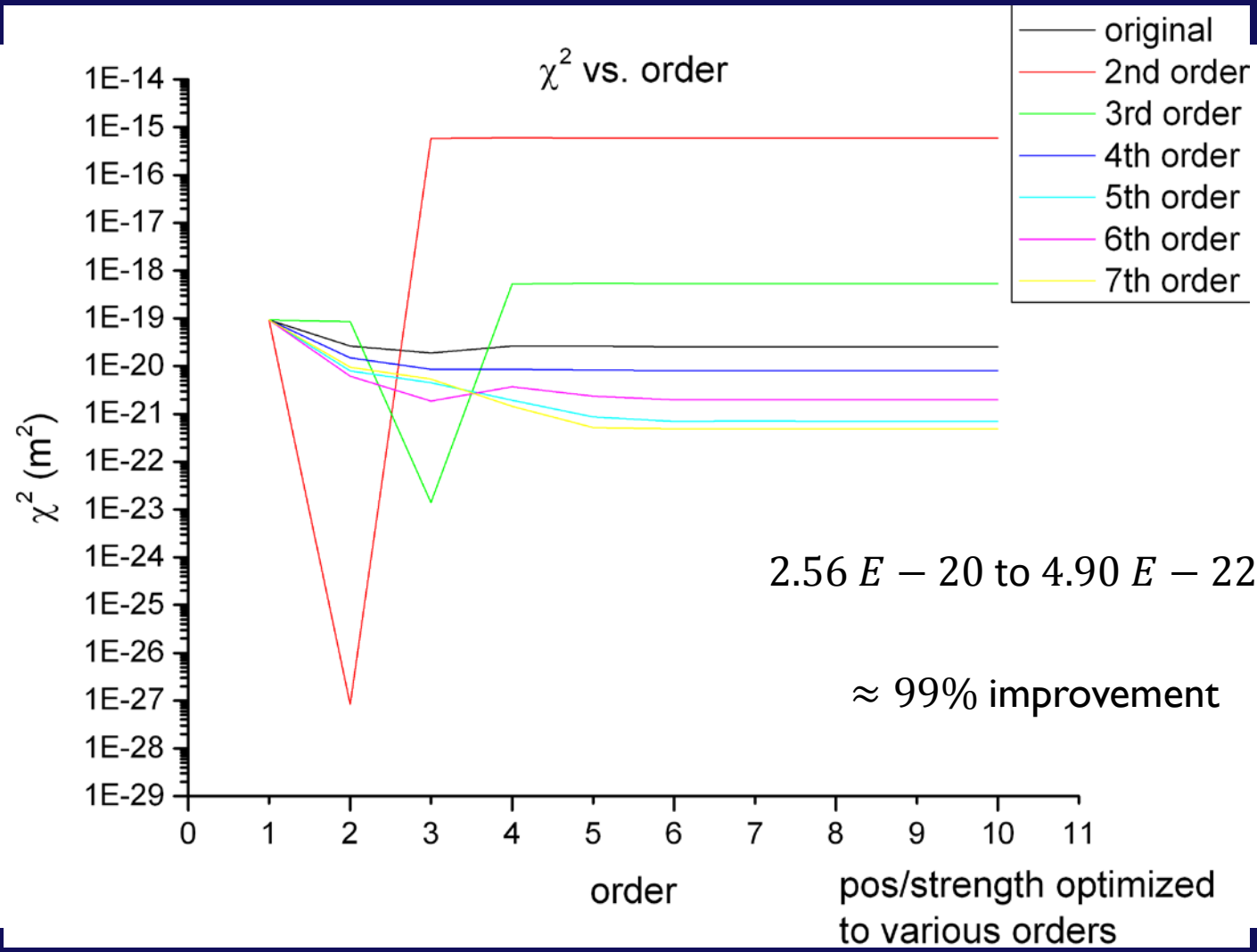
   σ_x σ_y χ^2

3. Choose a starting simplex:

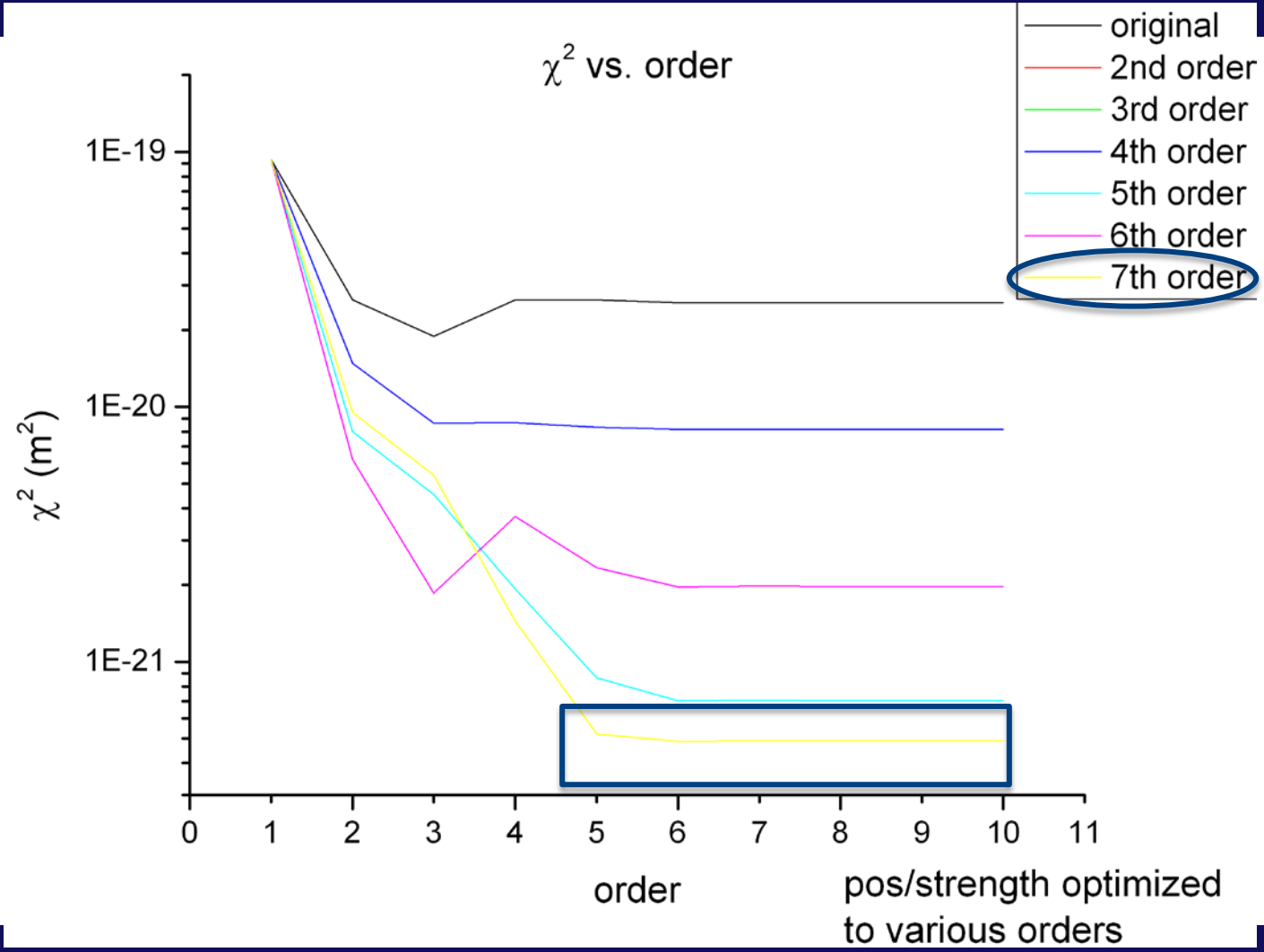
   $[0] * 14$ $[100, -100, 50] * 3$ $[0.1, -0.1, 0.1, -0.1, 0.1]$

4. Calculate σ_x, σ_y, or χ^2 for other orders.

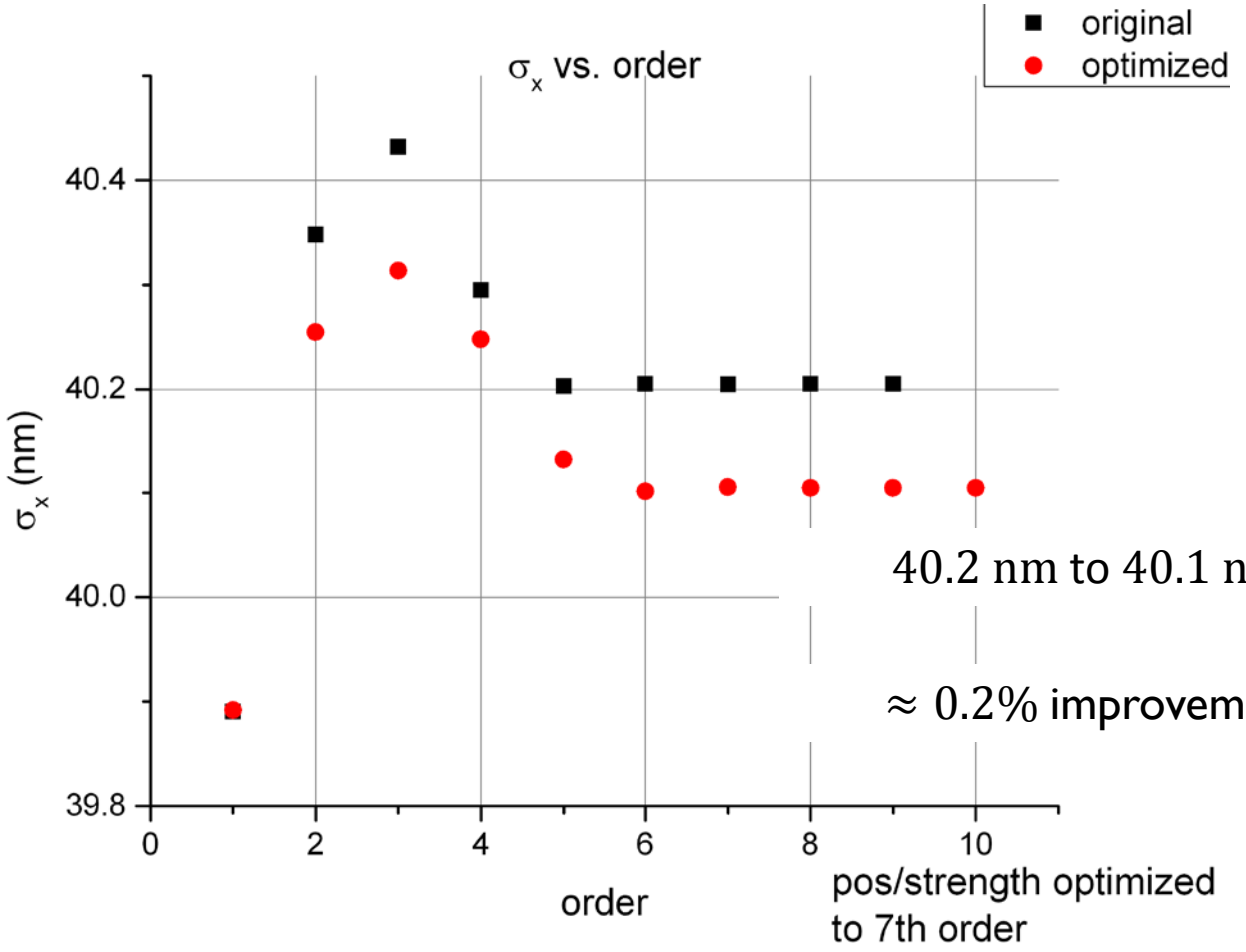
Optimization depends on order of calculation.



Optimization depends on order of calculation.



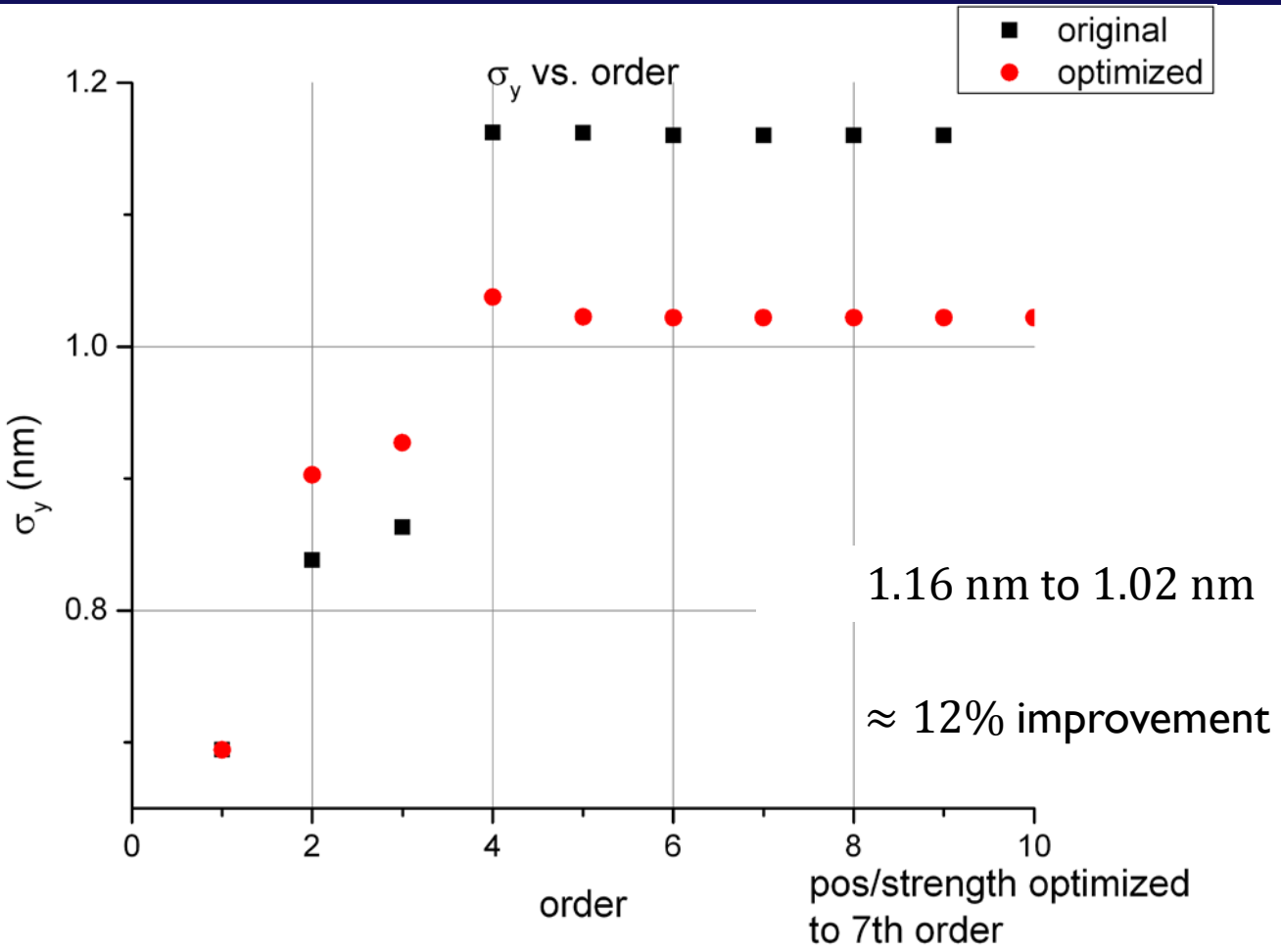
σ_x and σ_y for optimized χ^2



40.2 nm to 40.1 nm

≈ 0.2% improvement

σ_x and σ_y for optimized χ^2



Results

Element #	Nominal Lattice ($1/m^2$)	Improved Strength	Position change (m)
24	10.335585192	10.2881885328145	0.07967687682559
30	0.0	133.508521888545	0
31	89.66349022	-109.469292468306	0
32	0.0	68.8207501444652	0
38	-5.033440174	-5.09437310259855	0.049979477525608
43	15.196820336	15.5187207629783	0.052613241375277
49	0.0	-1.27849821750552	0
74	-6.053493698	-6.01584279027468	0.193170454943335
82	21.78589514	21.8375550635217	0.00738902713769

Summary of important scripts

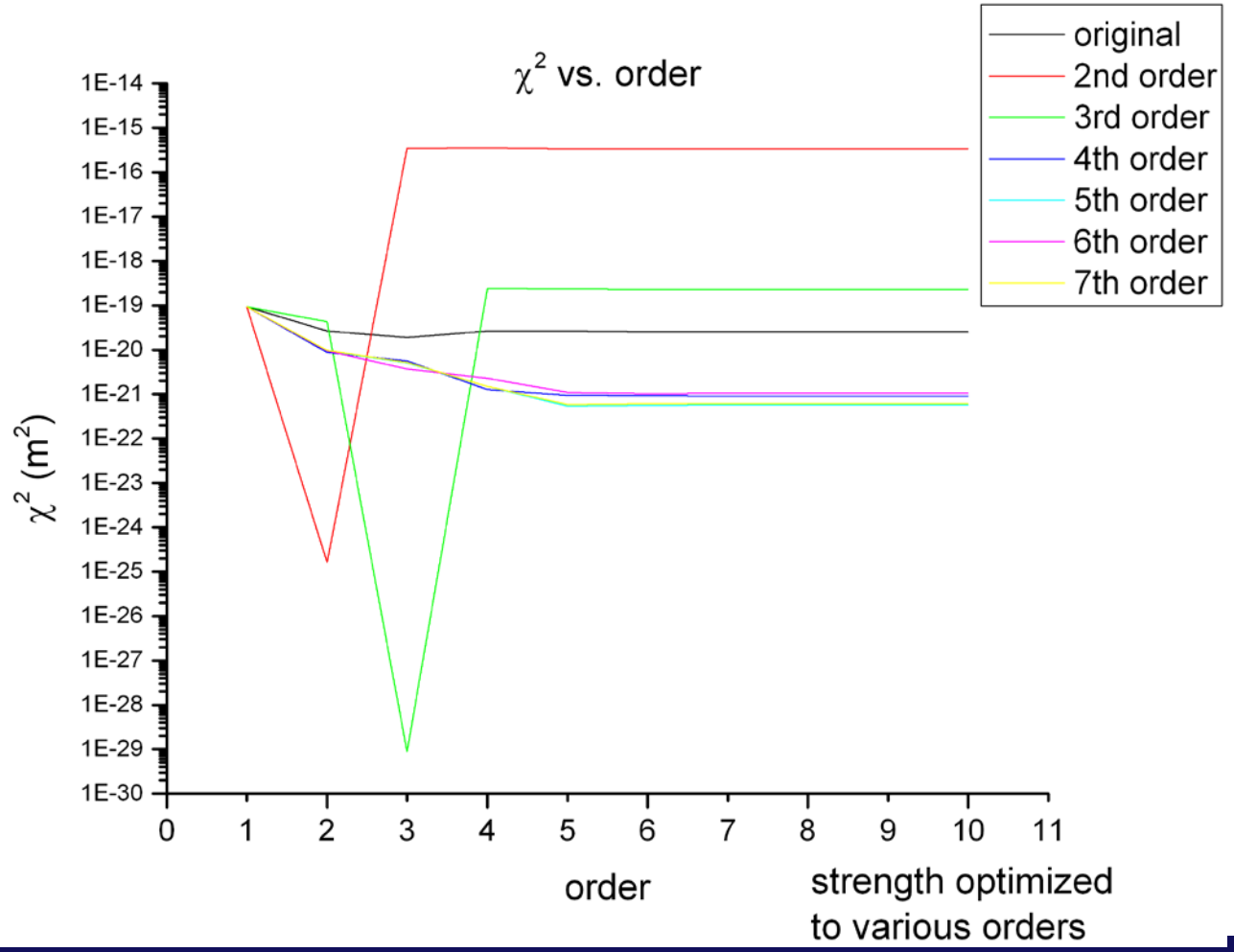
- **simplexduo.py** : minimizes χ^2 by adjusting all strengths and all positions
- **simplex5.py** : just for strength
- **simplex5pos.py** : just for position
- **chicalcduo.py** : calculate χ^2 with adjusted strengths/positions at any order
- **modsimplex.py** : "improved" simplex method; very dependent on initial guess

Future work

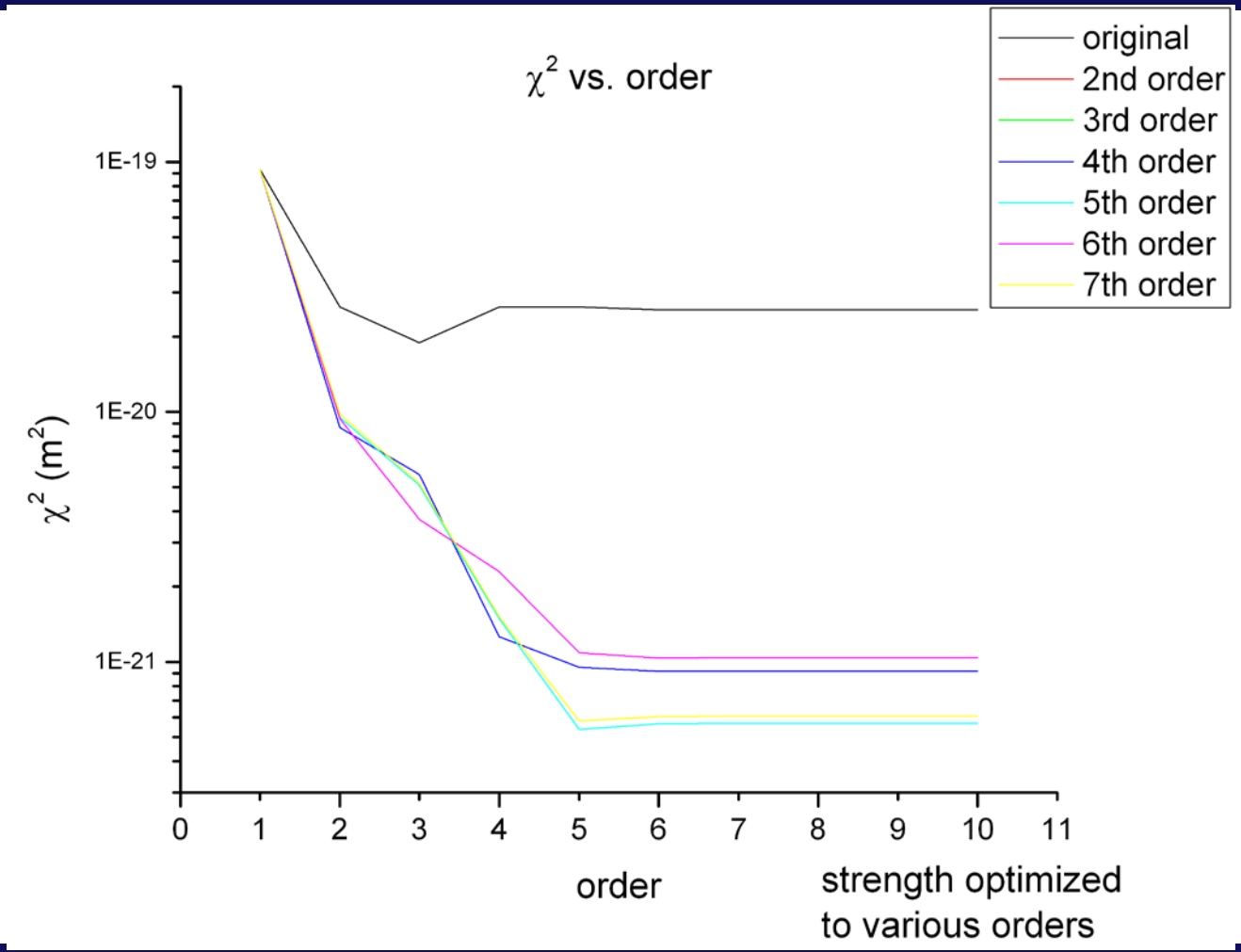
- more widespread use: add quadrupoles, use for other accelerators
- improve modsimplex.py to be used from the beginning (modification of gradient/guess calculation)
- 0^{th} order σ_x, σ_y given incorrectly in script
- impose more limits as needed

Back up slides

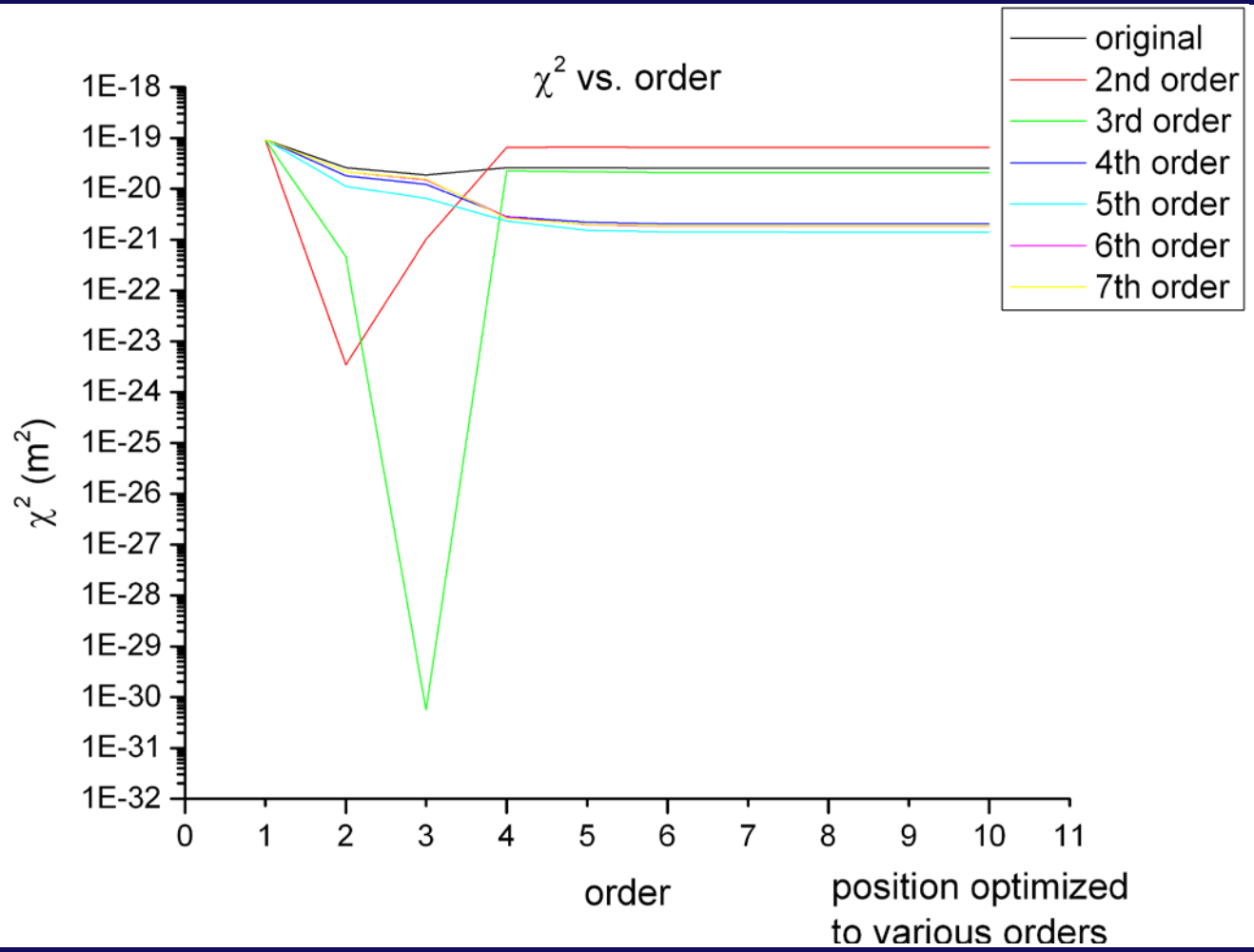
minimized χ^2 for optimized strengths



minimized χ^2 for optimized strengths



minimized χ^2 for optimized positions



minimized χ^2 for optimized positions