Numerical tools for CC simulations and results in SPS

A. Alekou, N. Triantafyllou, H. Bartosik

Many thanks to Riccardo de Maria, A. Mereghetti and V. Olsen for helping with MAD-X, SixTrack and SixDesk

Layout

Layout

- Crab cavity (CC) kick

Layout

- Crab cavity (CC) kick
- How to install a CC in MAD-X; example plots

Layout

- Crab cavity (CC) kick
- How to install a CC in MAD-X; example plots
- How to install static and oscillating multipoles in MAD-X; example plots

Layout

- Crab cavity (CC) kick
- How to install a CC in MAD-X; example plots
- How to install static and oscillating multipoles in MAD-X; example plots
- How to run SixTrack: necessary files, commands, example scripts

Layout

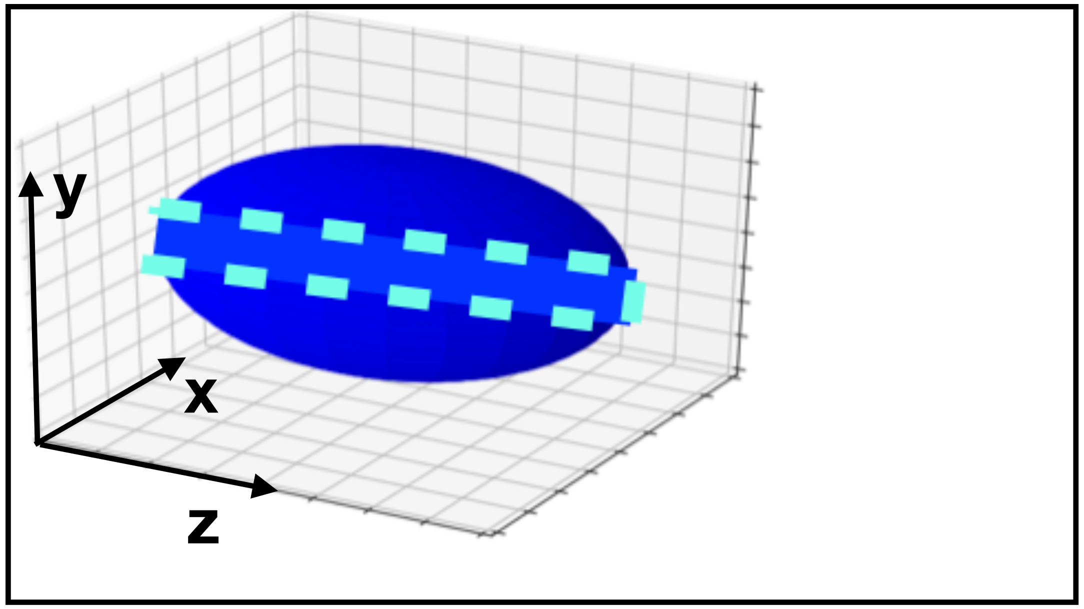
- Crab cavity (CC) kick
- How to install a CC in MAD-X; example plots
- How to install static and oscillating multipoles in MAD-X; example plots
- How to run SixTrack: necessary files, commands, example scripts
- How to dump a beam population in SixTrack

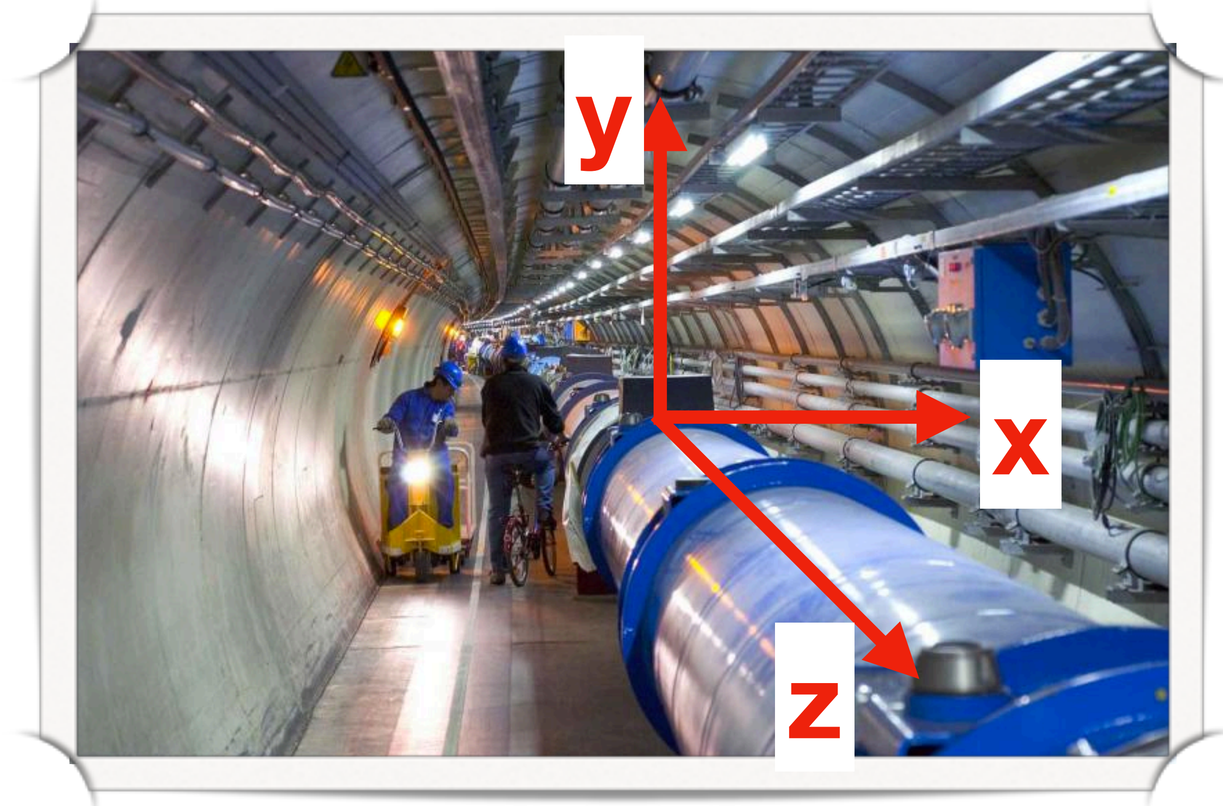
Layout

- Crab cavity (CC) kick
- How to install a CC in MAD-X; example plots
- How to install static and oscillating multipoles in MAD-X; example plots
- How to run SixTrack: necessary files, commands, example scripts
- How to dump a beam population in SixTrack
- How to slowly increase the V_{CC} in SixTrack

Layout

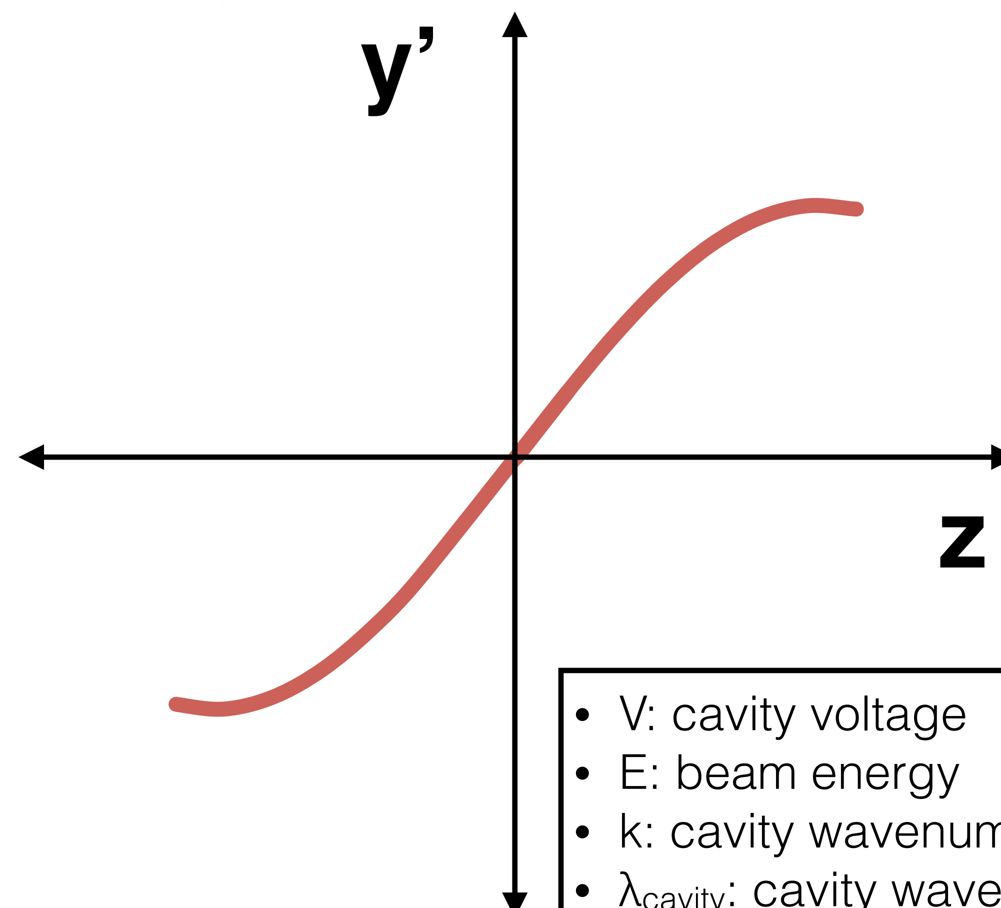
- Crab cavity (CC) kick
- How to install a CC in MAD-X; example plots
- How to install static and oscillating multipoles in MAD-X; example plots
- How to run SixTrack: necessary files, commands, example scripts
- How to dump a beam population in SixTrack
- How to slowly increase the V_{CC} in SixTrack
- DA simulations examples using SixDesk



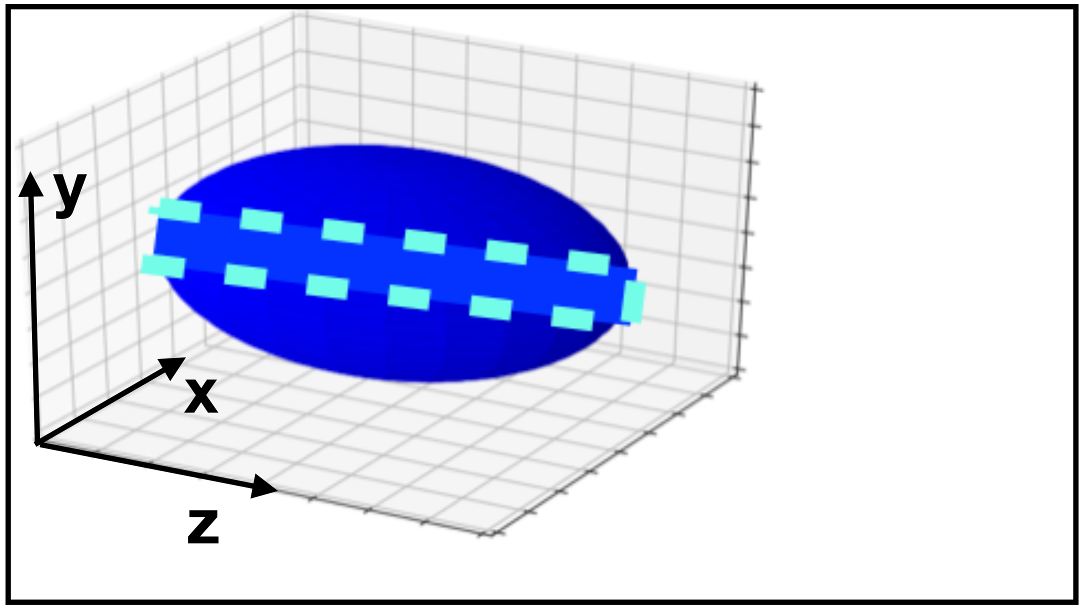


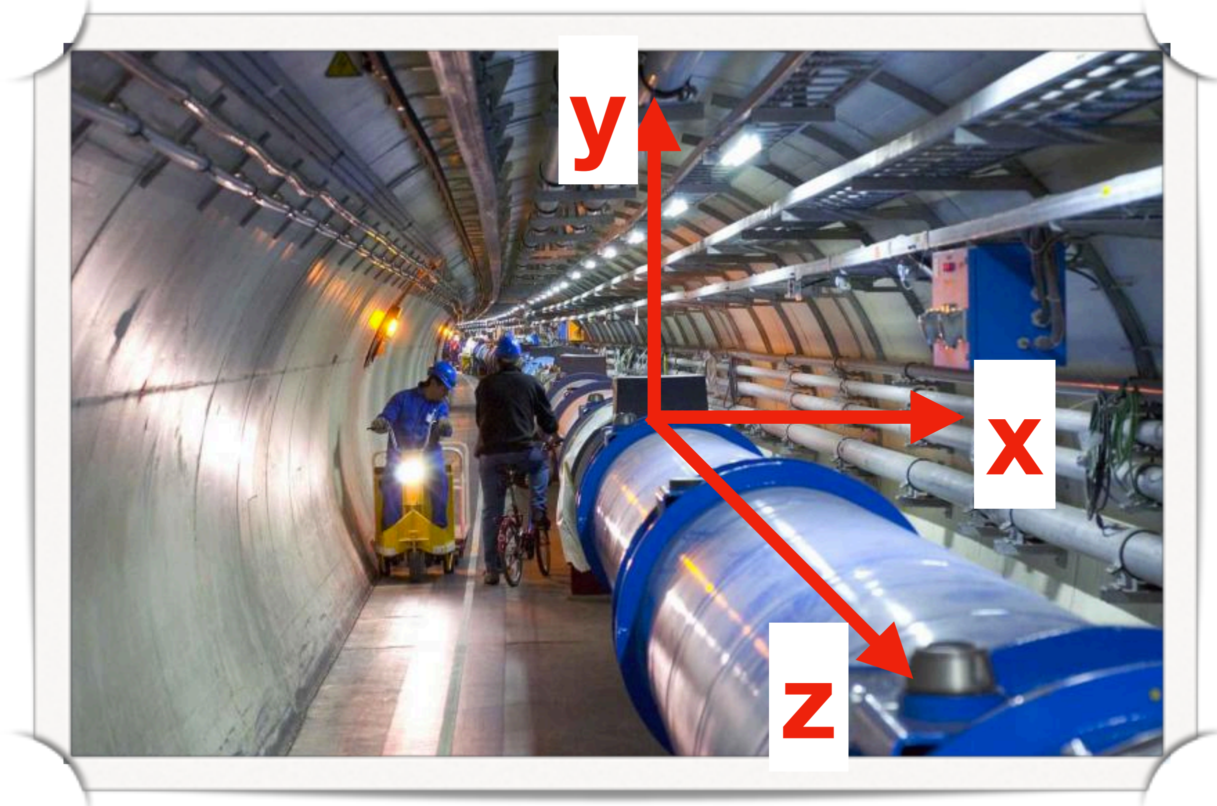
Crab Cavity kick:

$$y' = \frac{dy}{dz} = \frac{V}{E}\sin(kz + \phi)$$



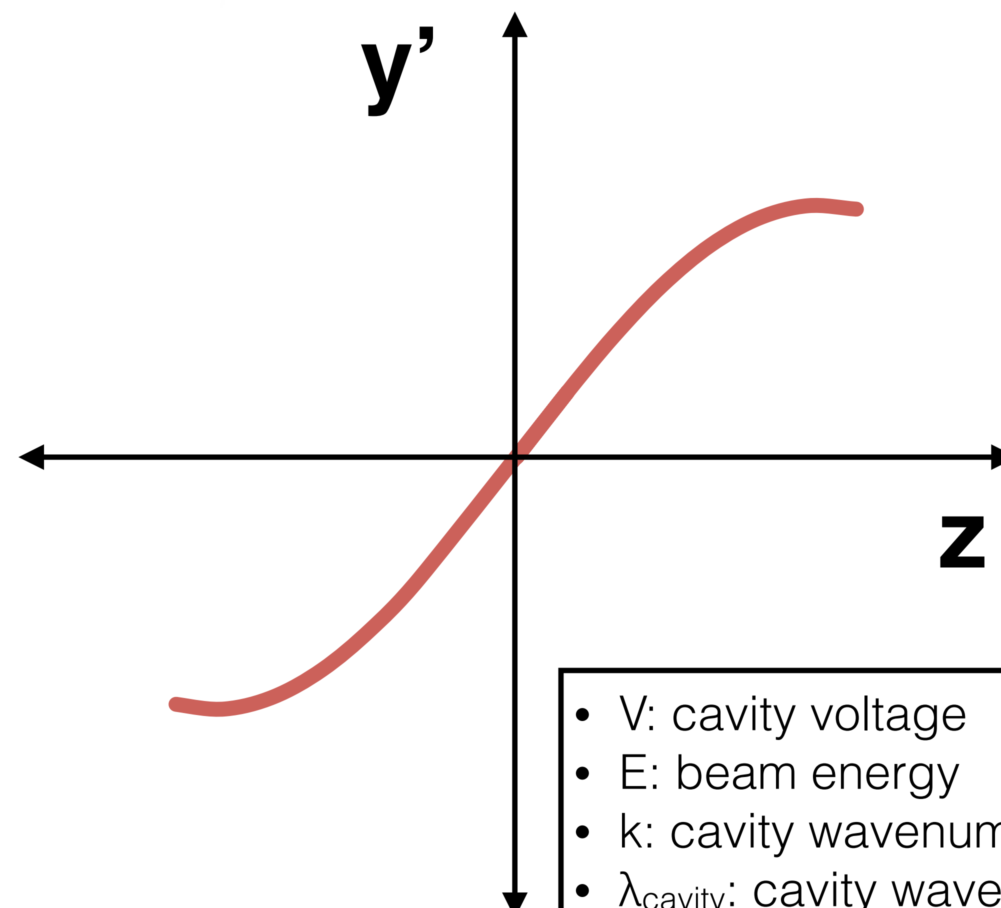
- V: cavity voltage
- E: beam energy
- k: cavity wavenumber ($2\pi/\lambda_{cavity}$)
- λ_{cavity}: cavity wavelength
- z: longitudinal position of particle
- ϕ: cavity phase



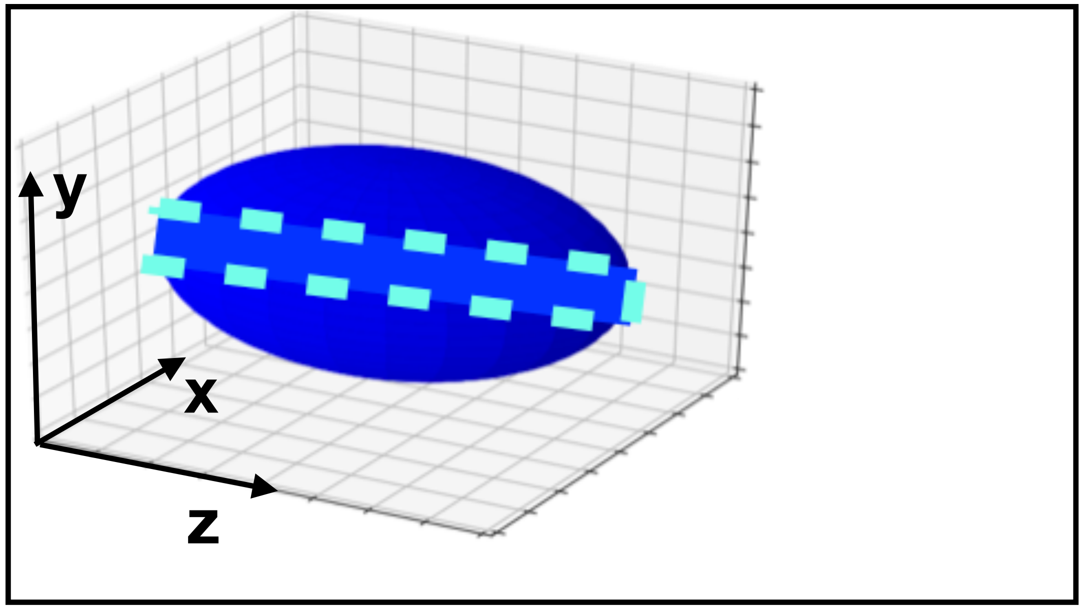


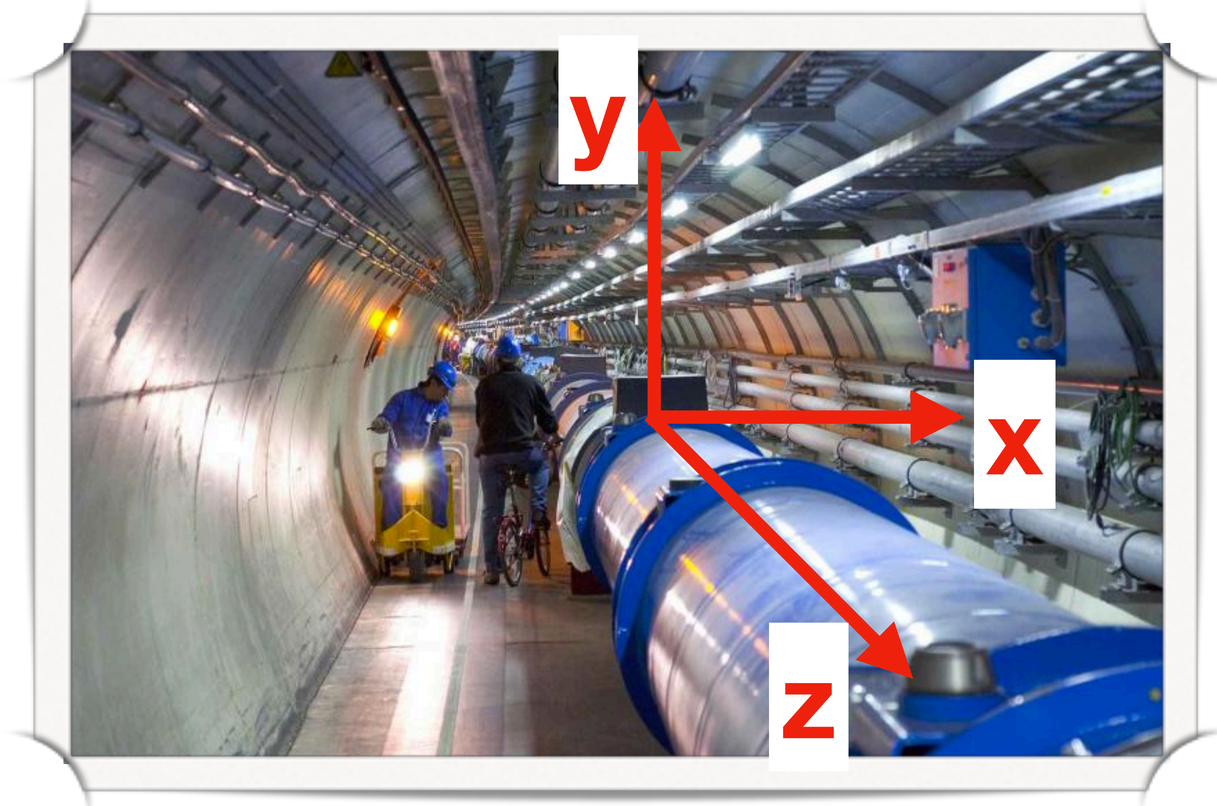
Crab Cavity kick:

$$y' = \frac{dy}{dz} = \frac{V}{E}\sin(kz + \phi)$$



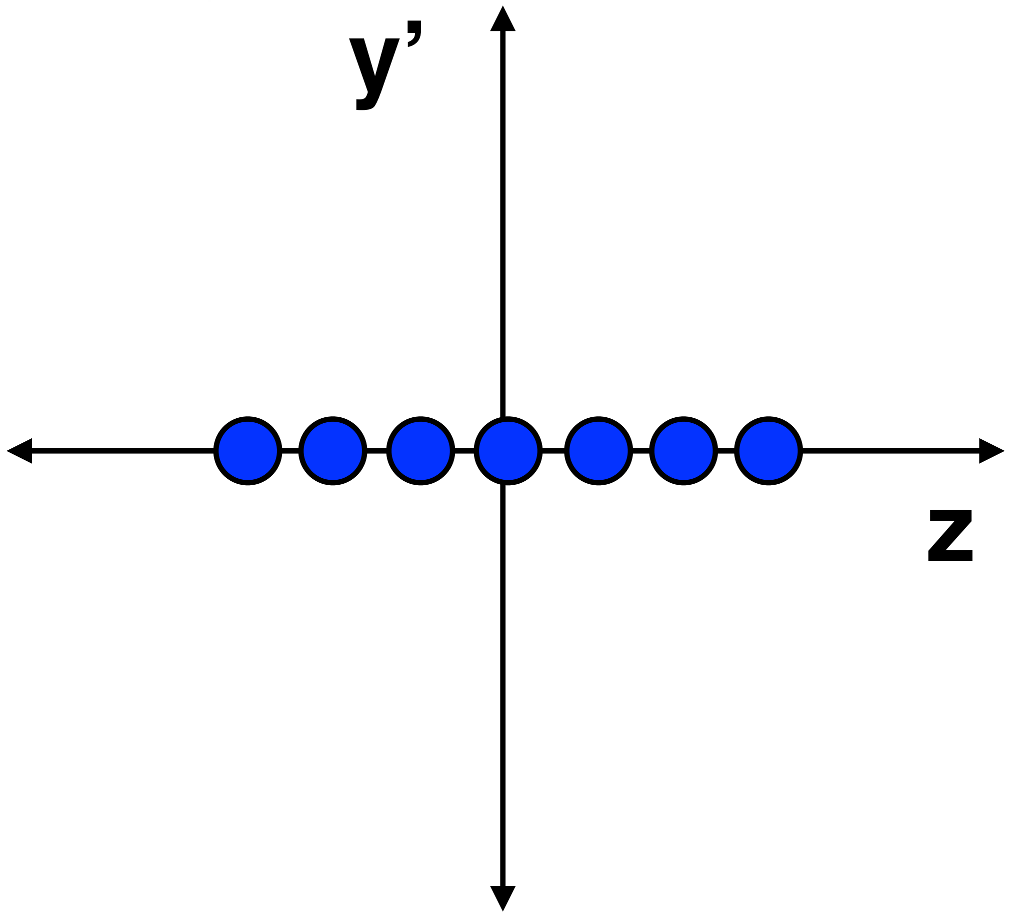
- V: cavity voltage
- E: beam energy
- k: cavity wavenumber ($2\pi/\lambda_{cavity}$)
- λ_{cavity}: cavity wavelength
- z: longitudinal position of particle
- ϕ: cavity phase

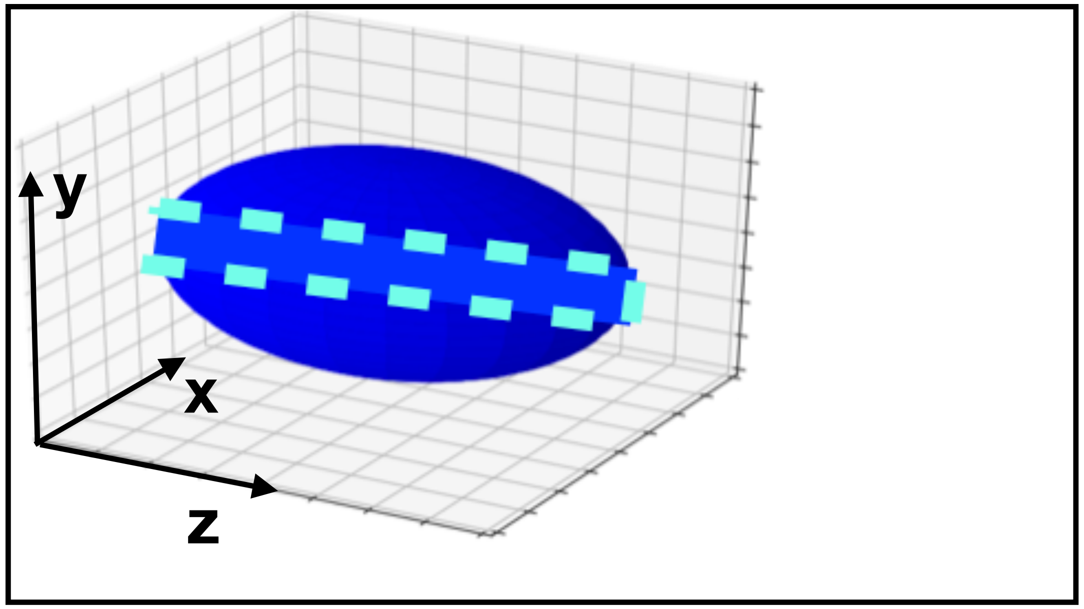


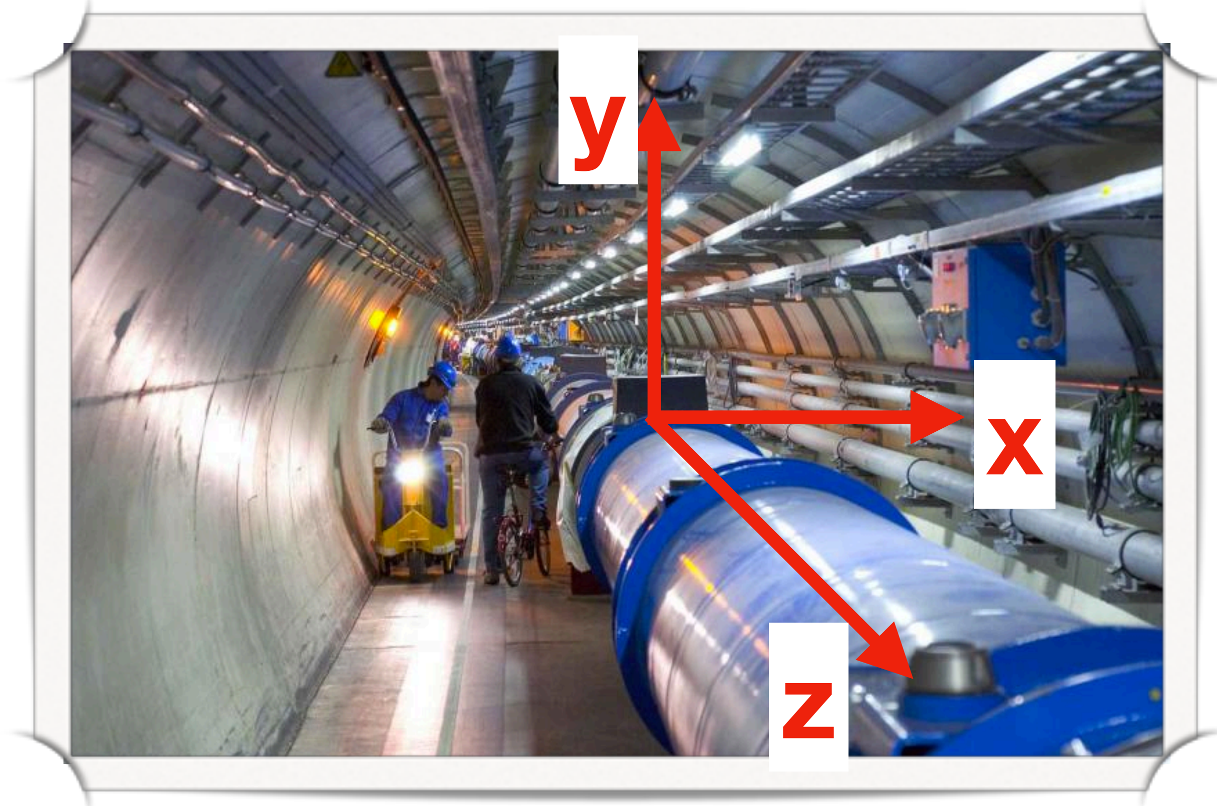


Crab Cavity kick:

$$y' = \frac{dy}{dz} = \frac{V}{E}\sin(kz + \phi)$$

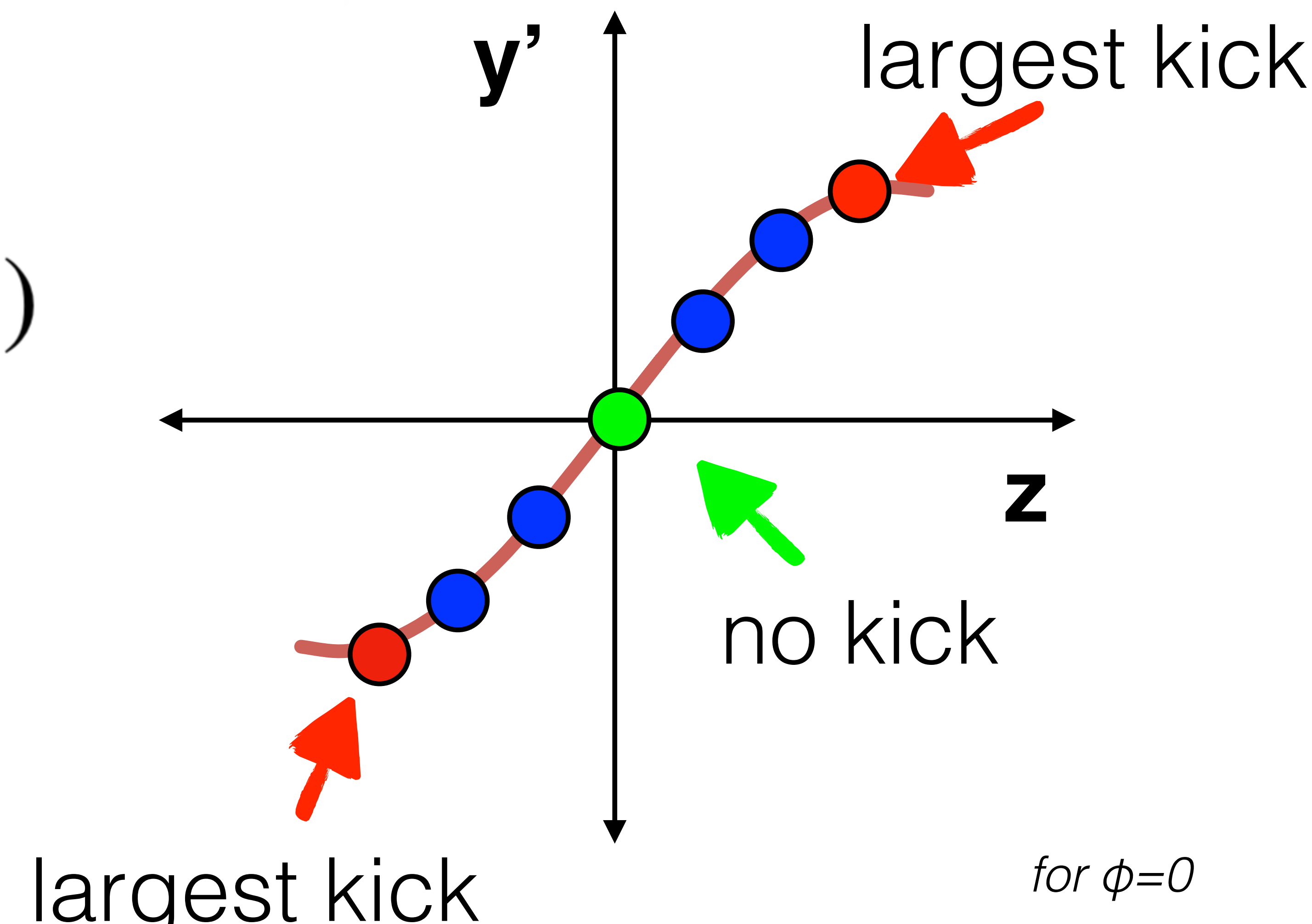






Crab Cavity kick:

$$y' = \frac{dy}{dz} = \frac{V}{E}\sin(kz + \phi)$$



Install CC in MAD-X

vertical CC

φ=180°

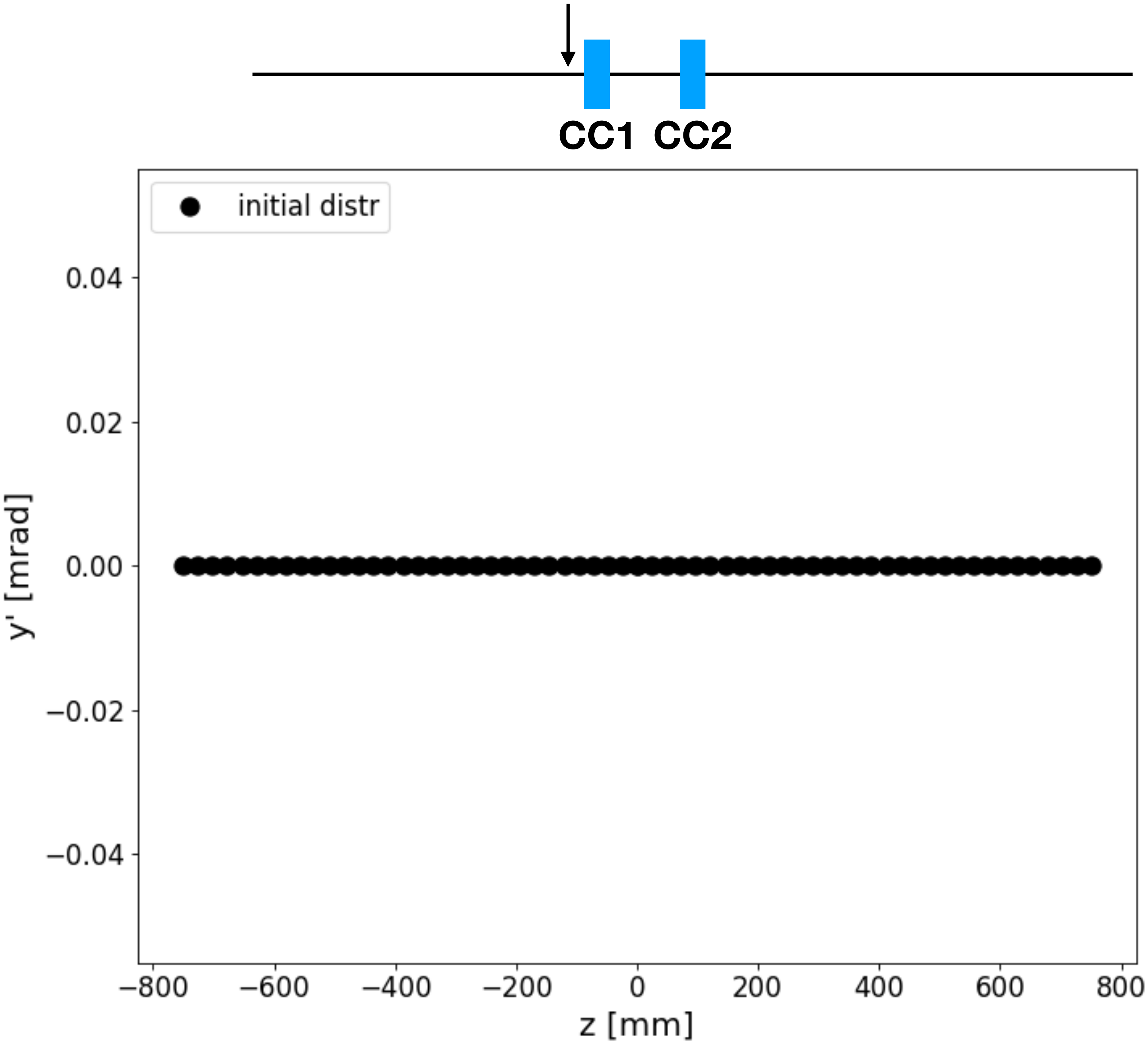
```
CRAVITY.1: CRABCAVITY, VOLT=2.0, FREQ=400, TILT=PI/2.;
CRAVITY.2: CRABCAVITY, VOLT=2.0, FREQ=400, TILT=PI/2., LAG=-0.5;
seqedit, sequence=sps;
INSTALL, ELEMENT=CRAVITY.1, AT=6312.7213;
INSTALL, ELEMENT=CRAVITY.2, AT=6313.3213;
endedit;
```

given as input in CC description

$$y' = \frac{V}{E} \sin(kz + \phi)$$

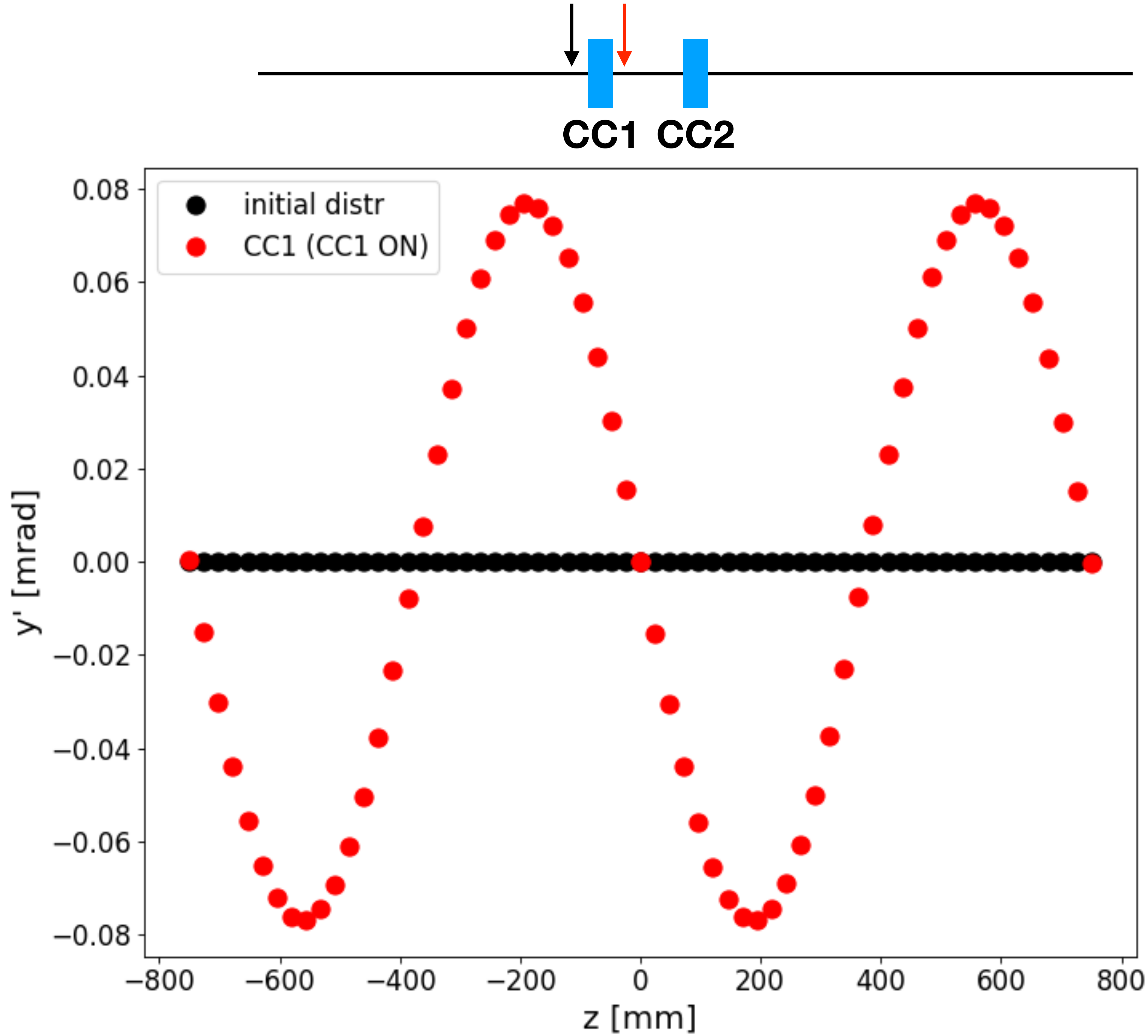
taken by mad-x script

RF phase: default is 0, CC has no effect at z=0



x=x’=y=y’=0

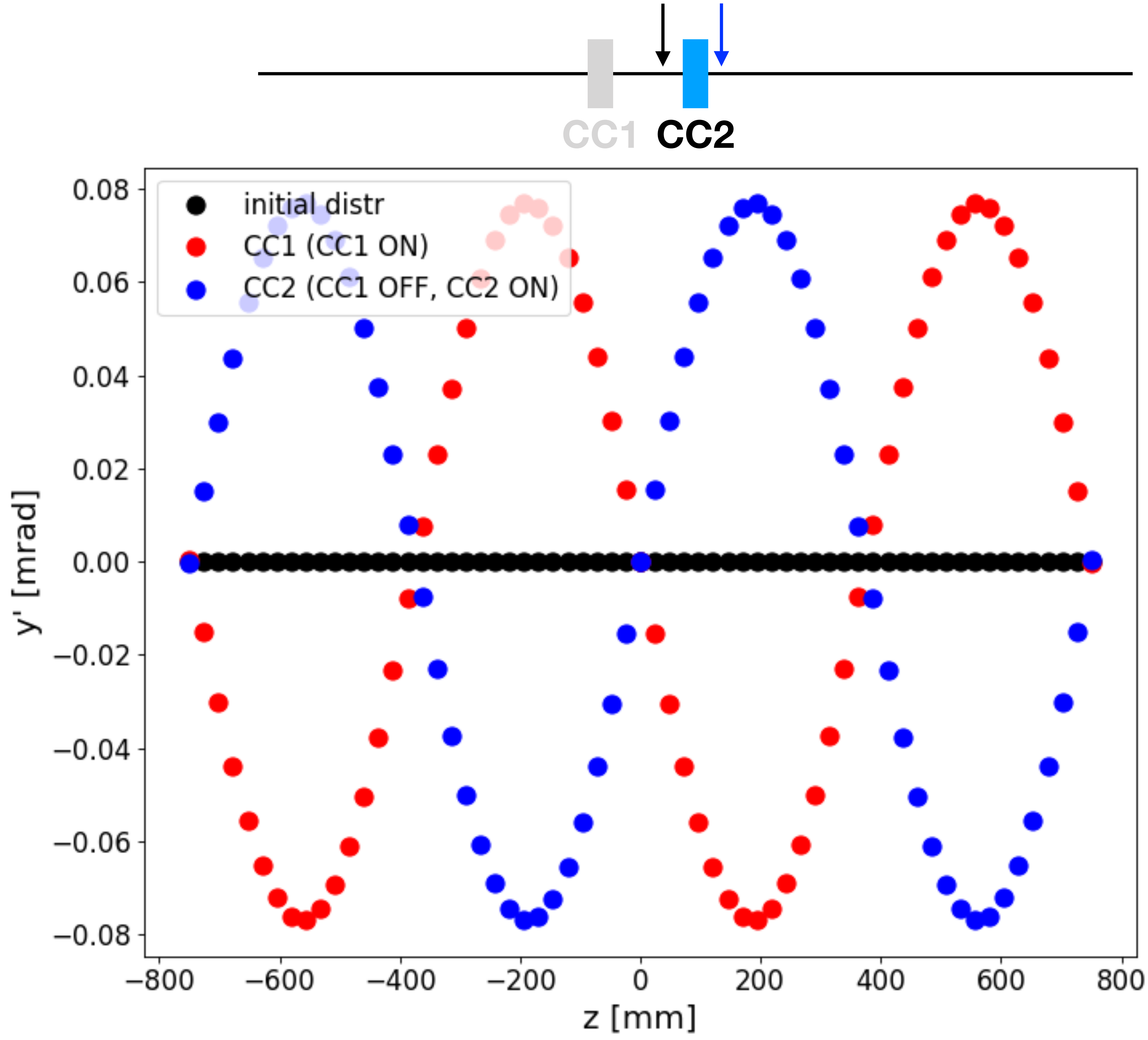
Plotting tip: <u>Veronica’s code</u> on GitHub



x=x'=y=y'=0

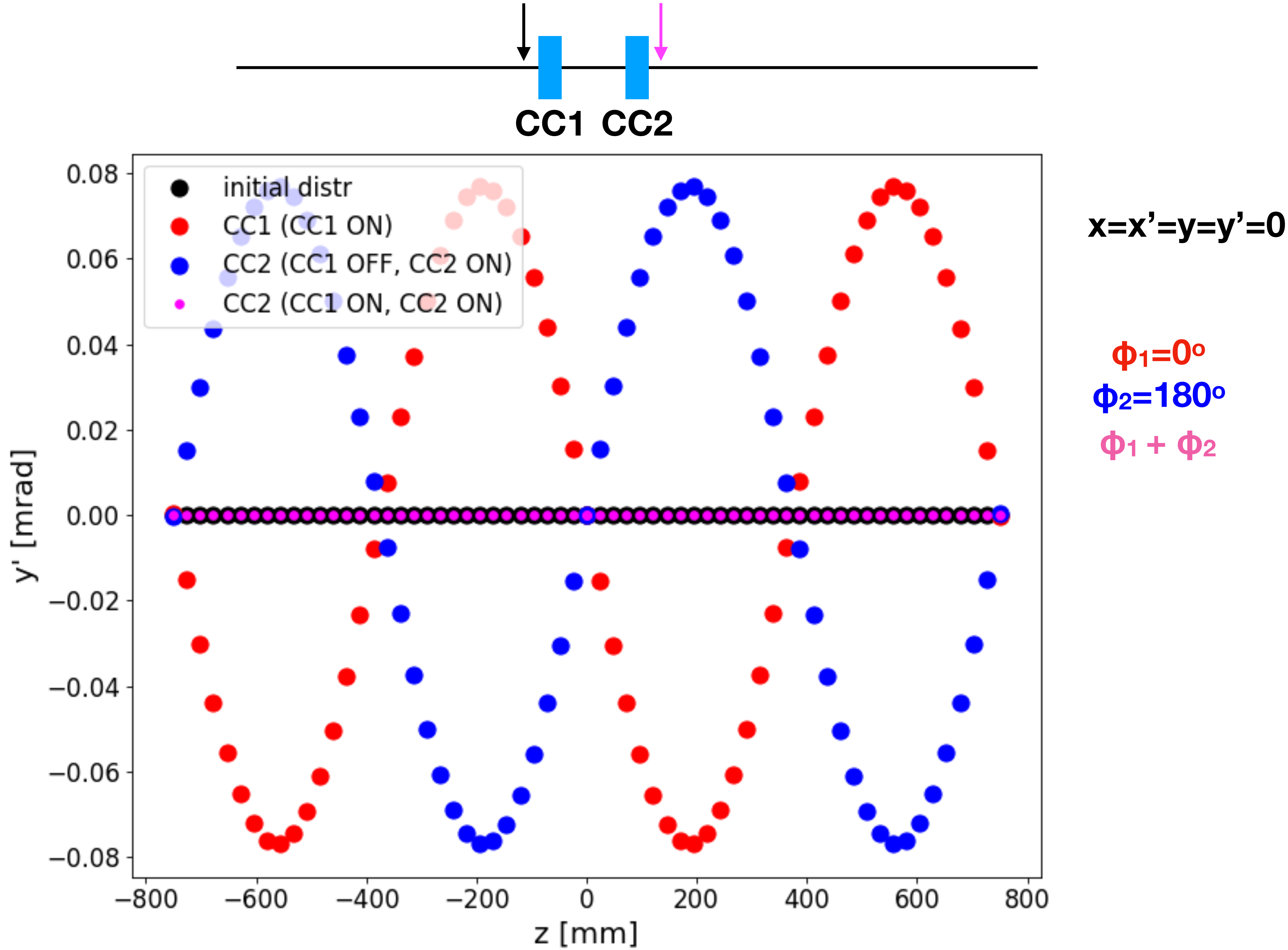
ϕ_1=0°

f=400MHz, λ=750 mm



x=x'=y=y'=0

$\phi_1=0^o$
$\phi_2=180^o$



CC1 CC2
initial distr
CC1 (CC1 ON)
CC2 (CC1 OFF, CC2 ON)
CC2 (CC1 ON, CC2 ON)
y' [mrad]
z [mm]
x=x'=y=y'=0
Φ1=0°
Φ2=180°
Φ1 + Φ2

Static vs oscillating (RF) multipole

Static vs oscillating (RF) multipole

- **n**ormal static quadrupolar multipole:
  `MULT.1: MULTIPOLE, KNL={0,multStrength};`

Static vs oscillating (RF) multipole

- **n**ormal static quadrupolar multipole:
  `MULT.1: MULTIPOLE, KNL={0,multStrength};`

- **n**ormal oscillating quadrupolar multipole:
  `MULT.1: RFMULTIPOLE, FREQ=400., KNL={0,multStrength}, PNL={0,0.25};`

`multStrength`: *already normalised with energy*

$$y' = \frac{V}{E}\cos(kz + \phi)$$

given as input in MULT description

Static vs oscillating (RF) multipole

- **n**ormal static quadrupolar multipole:
  `MULT.1: MULTIPOLE, KNL={0,multStrength};`

- **n**ormal oscillating quadrupolar multipole:
  `MULT.1: RFMULTIPOLE, FREQ=400., KNL={0,multStrength}, PNL={0,0.25};`

`multStrength`: *already normalised with energy*

$$y' = \frac{V}{E}\cos(kz + \phi)$$

given as input in MULT description

Note: PNL, PSL: RF multipole phase; set to 0.25 (π/2) if you want multipole to have no effect at z=0

Static vs oscillating (RF) multipole

- **n**ormal static quadrupolar multipole:
  `MULT.1: MULTIPOLE, KNL={0,multStrength};`

- **n**ormal oscillating quadrupolar multipole:
  `MULT.1: RFMULTIPOLE, FREQ=400., KNL={0,multStrength}, PNL={0,0.25};`

- **s**kewed oscillating sextupolar multipole:
  `MULT.1, FREQ=400., KSL={0,0,multStrength}, PSL={0,0,0.25};`

`multStrength`*: already normalised with energy*

$$y' = \frac{V}{E}\cos(kz + \phi)$$

given as input in MULT description

Note: PNL, PSL: RF multipole phase; set to 0.25 (π/2) if you want multipole to have no effect at z=0

Static vs oscillating (RF) multipole

- **n**ormal static quadrupolar multipole:
  `MULT.1: MULTIPOLE, KNL={0,multStrength};`

- **n**ormal oscillating quadrupolar multipole:
  `MULT.1: RFMULTIPOLE, FREQ=400., KNL={0,multStrength}, PNL={0,0.25};`

- **s**kewed oscillating sextupolar multipole:
  `MULT.1, FREQ=400., KSL={0,0,multStrength}, PSL={0,0,0.25};`

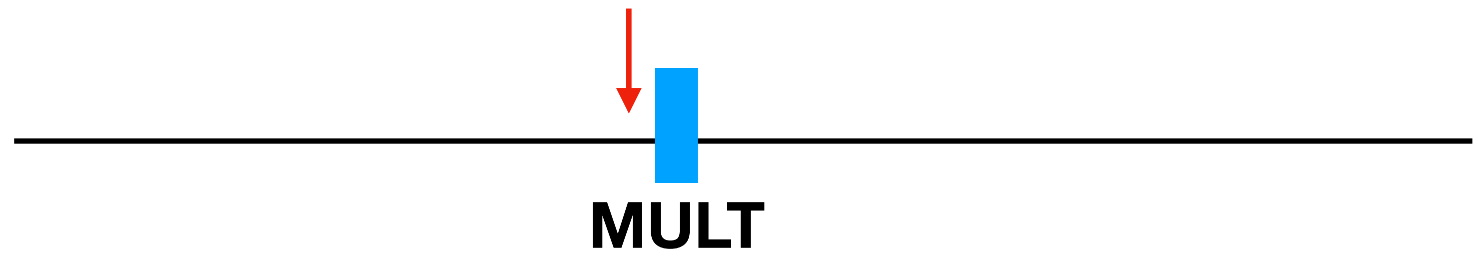
- **n**ormal oscillating quadrupolar multipole **AND s**kewed oscillating sextupolar multipole:
  `MULT.1, FREQ=400., KNL={0,QmultStrength}, PNL={0,0.25}, KSL={0,0,SmultStrength}, PSL={0,0,0.25};`

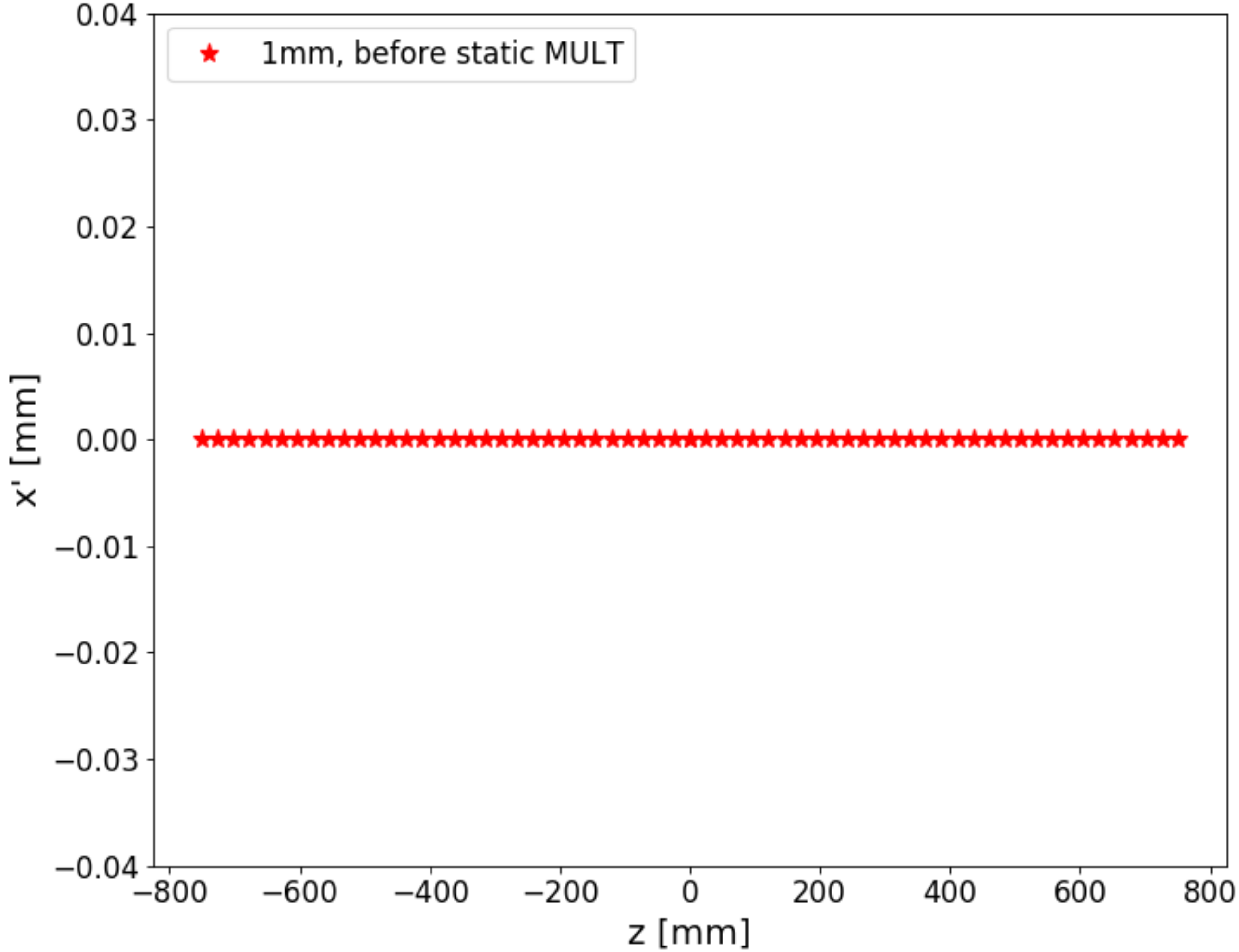
`multStrength`: *already normalised with energy*

$$y' = \frac{V}{E}\cos(kz + \phi)$$

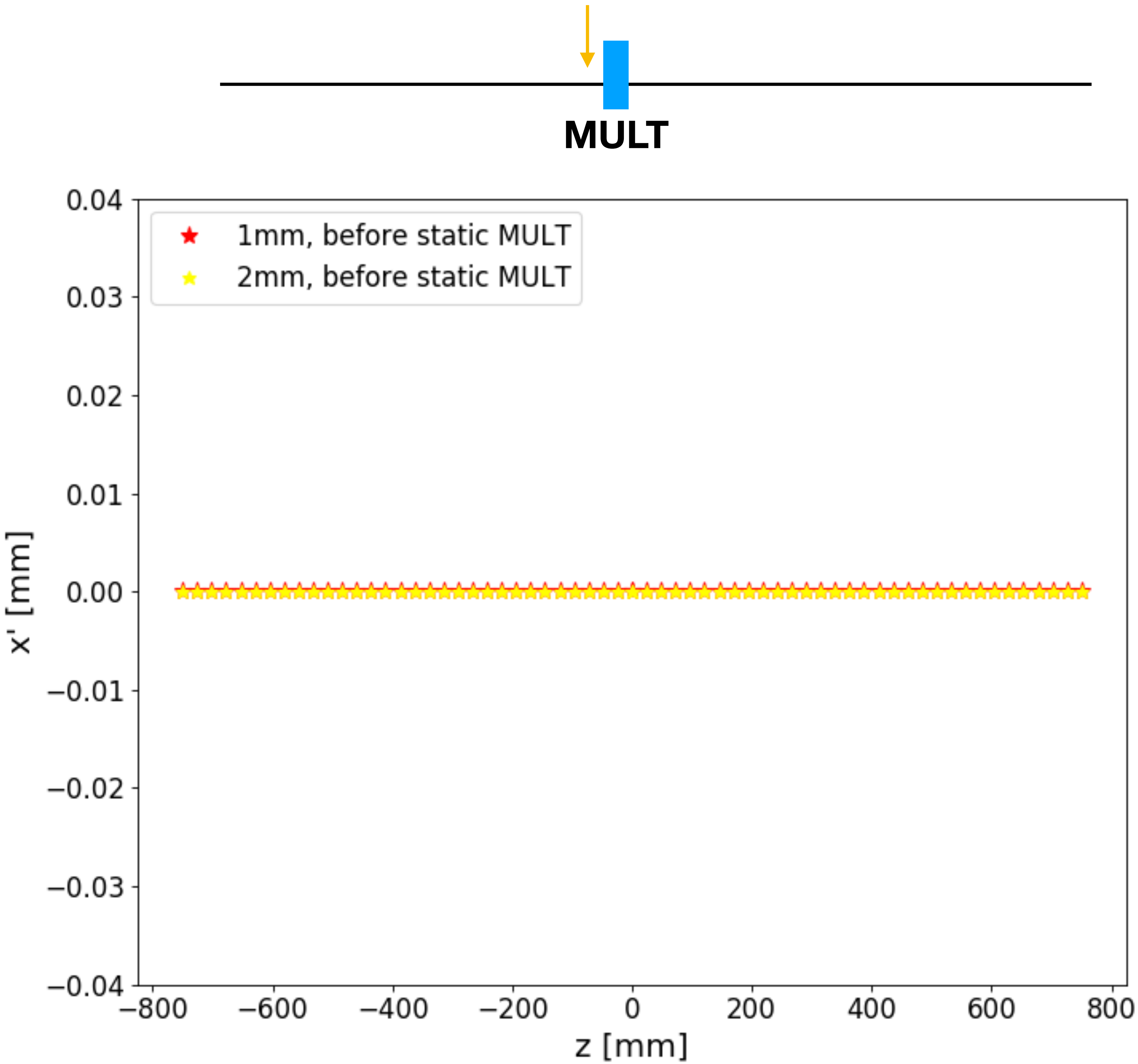
given as input in MULT description

Note: PNL, PSL: RF multipole phase; set to 0.25 (π/2) if you want multipole to have no effect at z=0


MULT

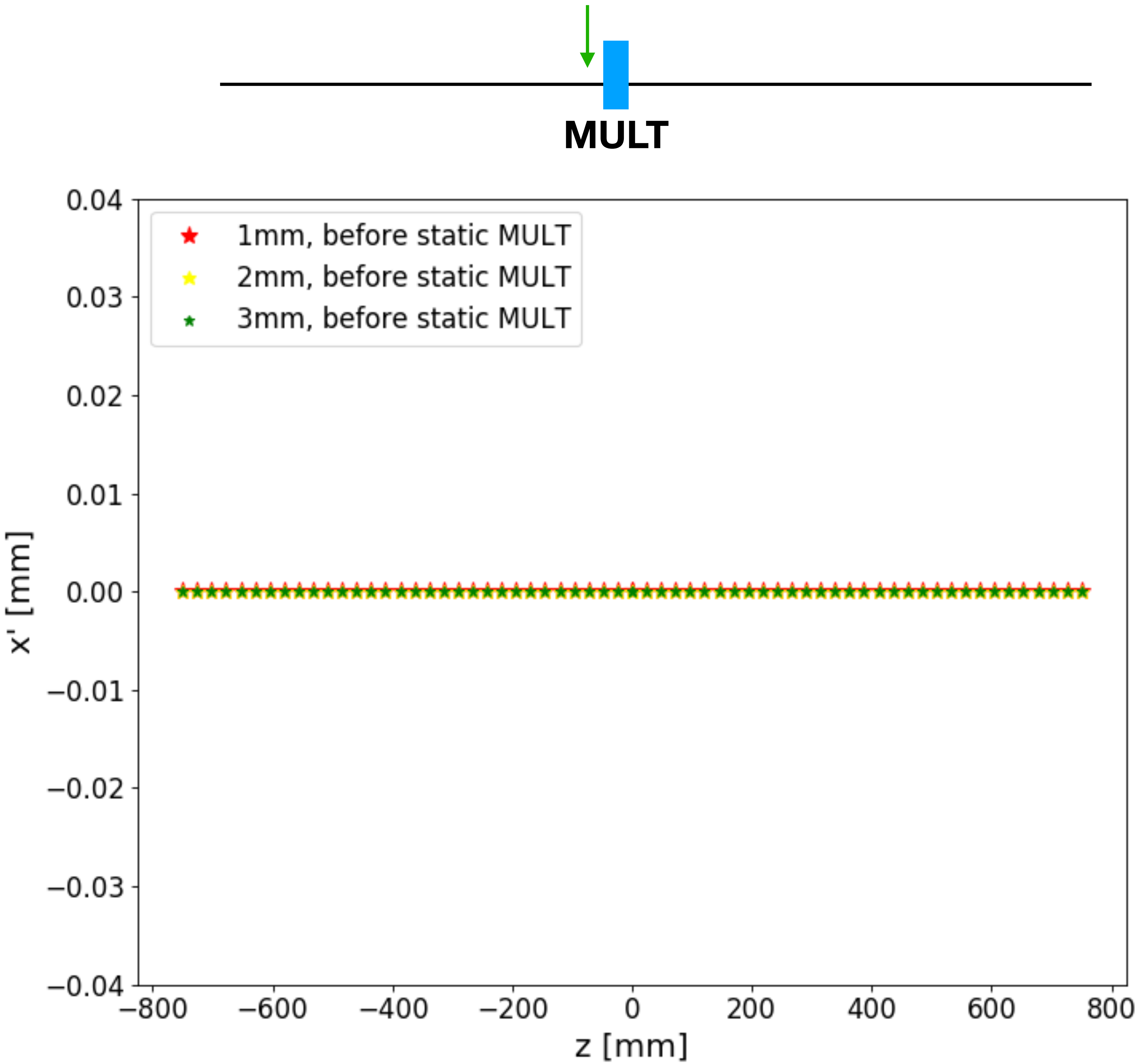

1mm, before static MULT
0.04
0.03
0.02
0.01
0.00
−0.01
−0.02
−0.03
−0.04
x' [mm]
−800
−600
−400
−200
0
200
400
600
800
z [mm]

x=1 mm
x’=y=y’=0



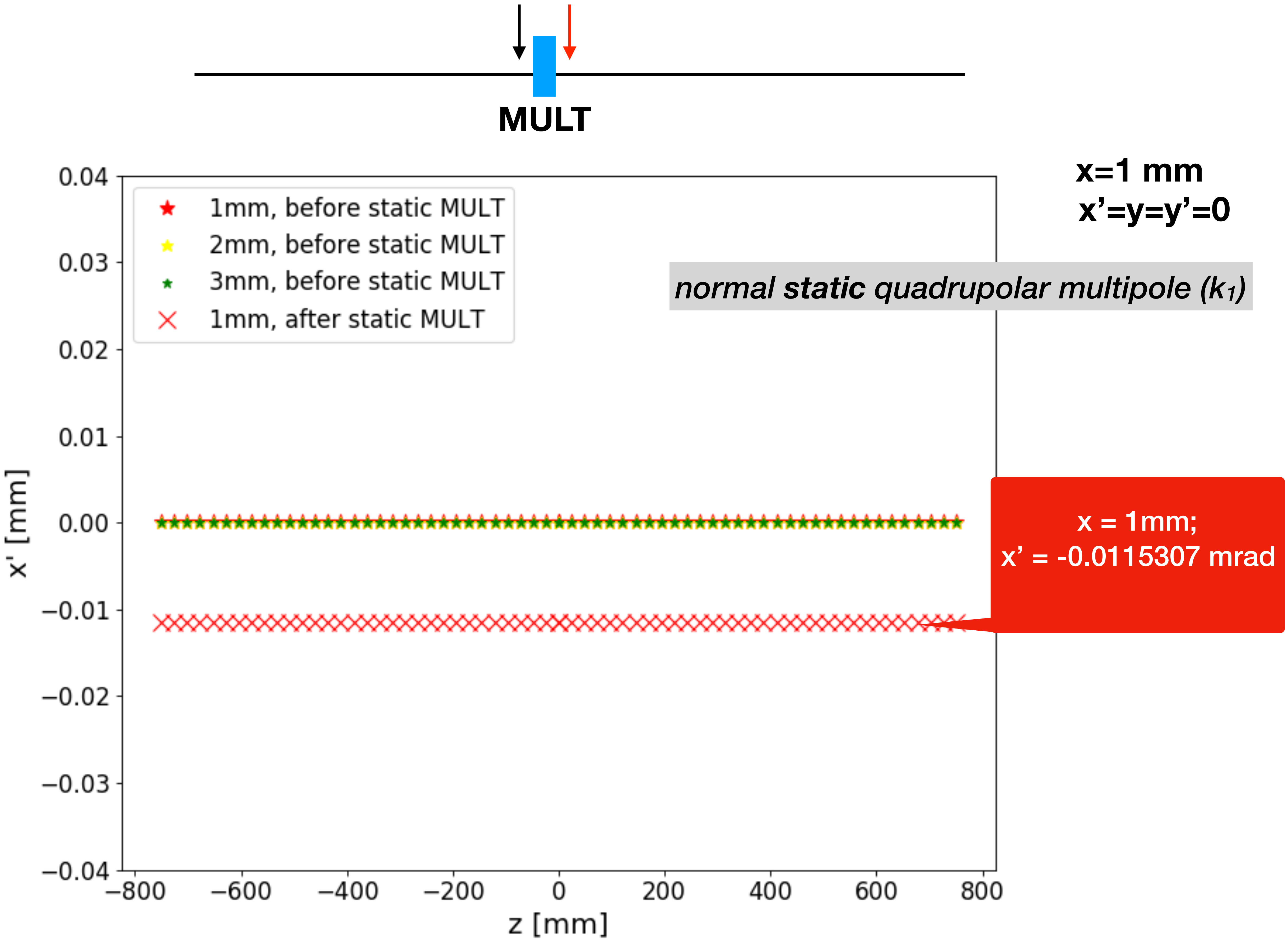
MULT
1mm, before static MULT
2mm, before static MULT
0.04
0.03
0.02
0.01
0.00
−0.01
−0.02
−0.03
−0.04
x' [mm]
−800
−600
−400
−200
0
200
400
600
800
z [mm]

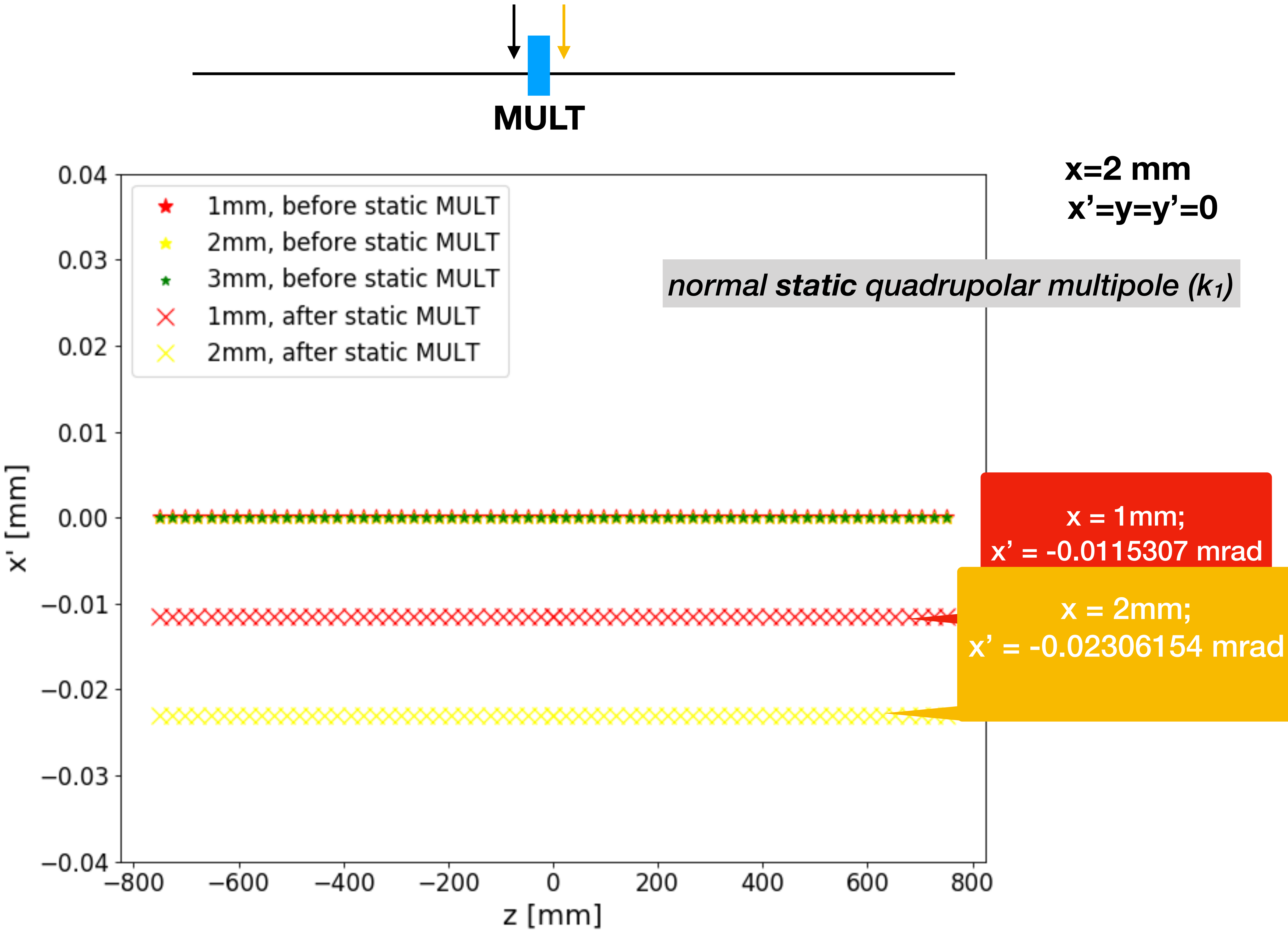
x=2 mm
x'=y=y'=0

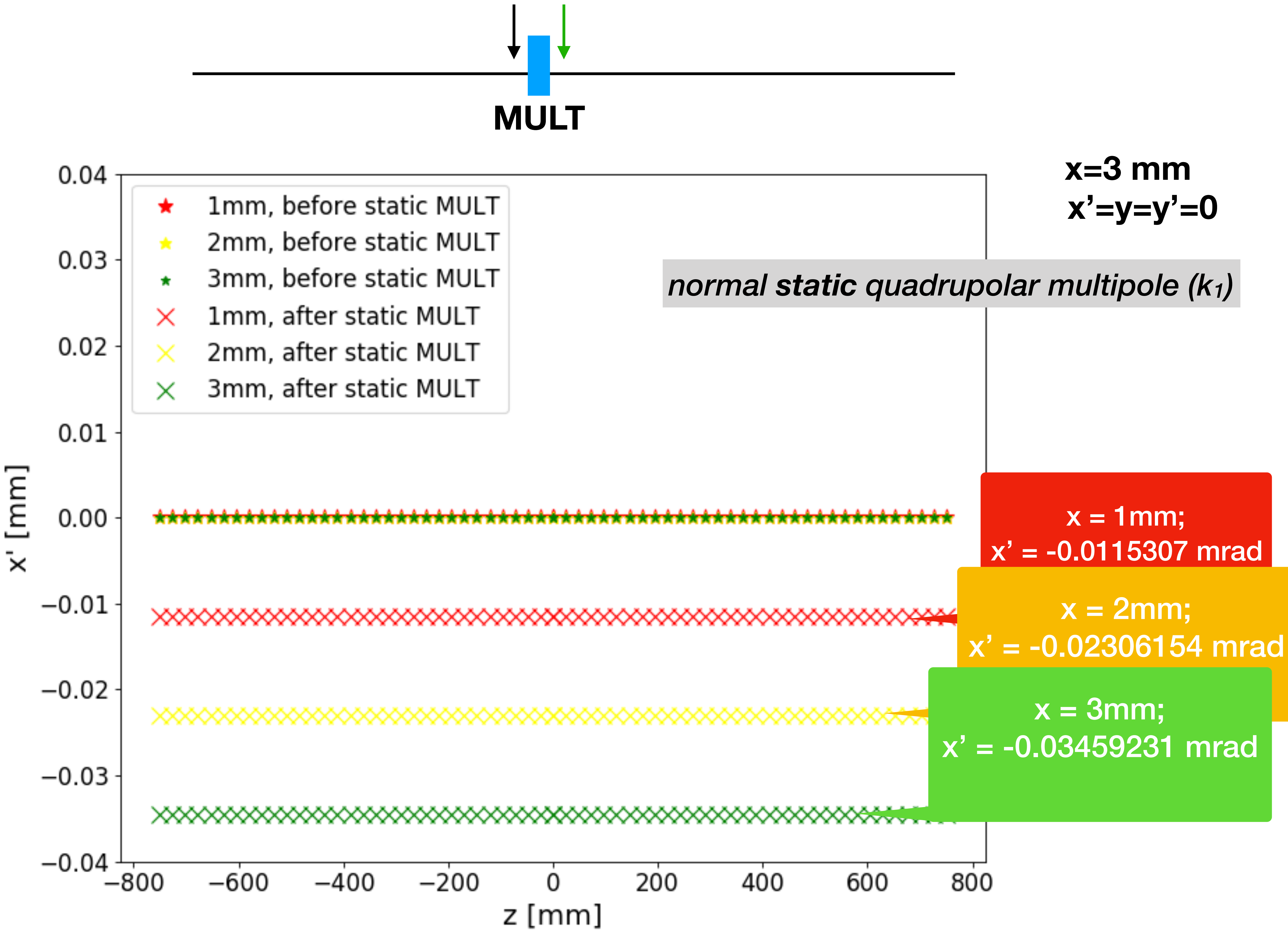

MULT
1mm, before static MULT
2mm, before static MULT
3mm, before static MULT
0.04
0.03
0.02
0.01
0.00
−0.01
−0.02
−0.03
−0.04
x' [mm]
−800
−600
−400
−200
0
200
400
600
800
z [mm]

x=3 mm

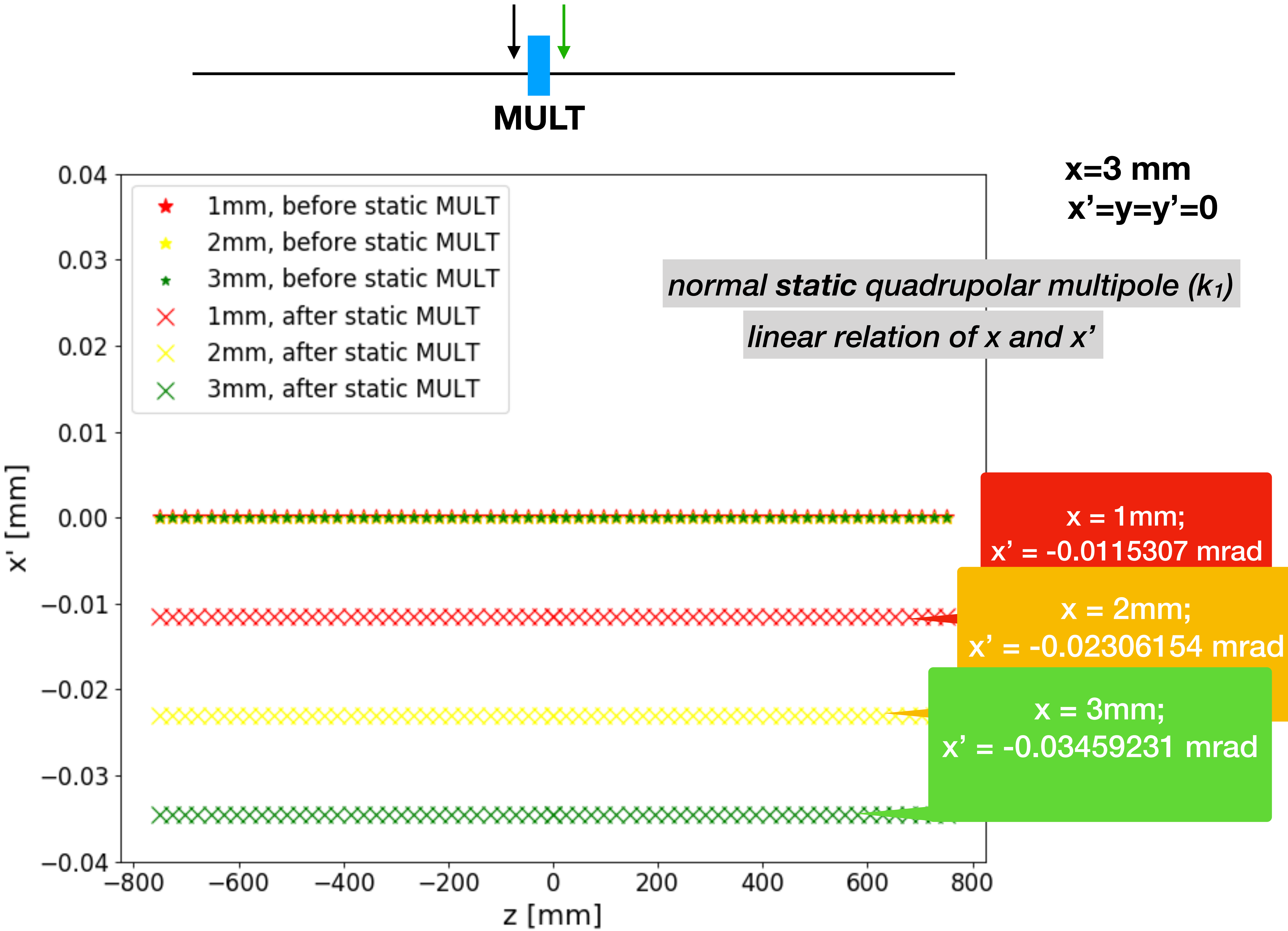
x’=y=y’=0


MULT
x=1 mm
x'=y=y'=0
normal static quadrupolar multipole (k_1)
1mm, before static MULT
2mm, before static MULT
3mm, before static MULT
1mm, after static MULT
x = 1mm;
x' = -0.0115307 mrad
x' [mm]
z [mm]
0.04
0.03
0.02
0.01
0.00
−0.01
−0.02
−0.03
−0.04
−800
−600
−400
−200
0
200
400
600
800

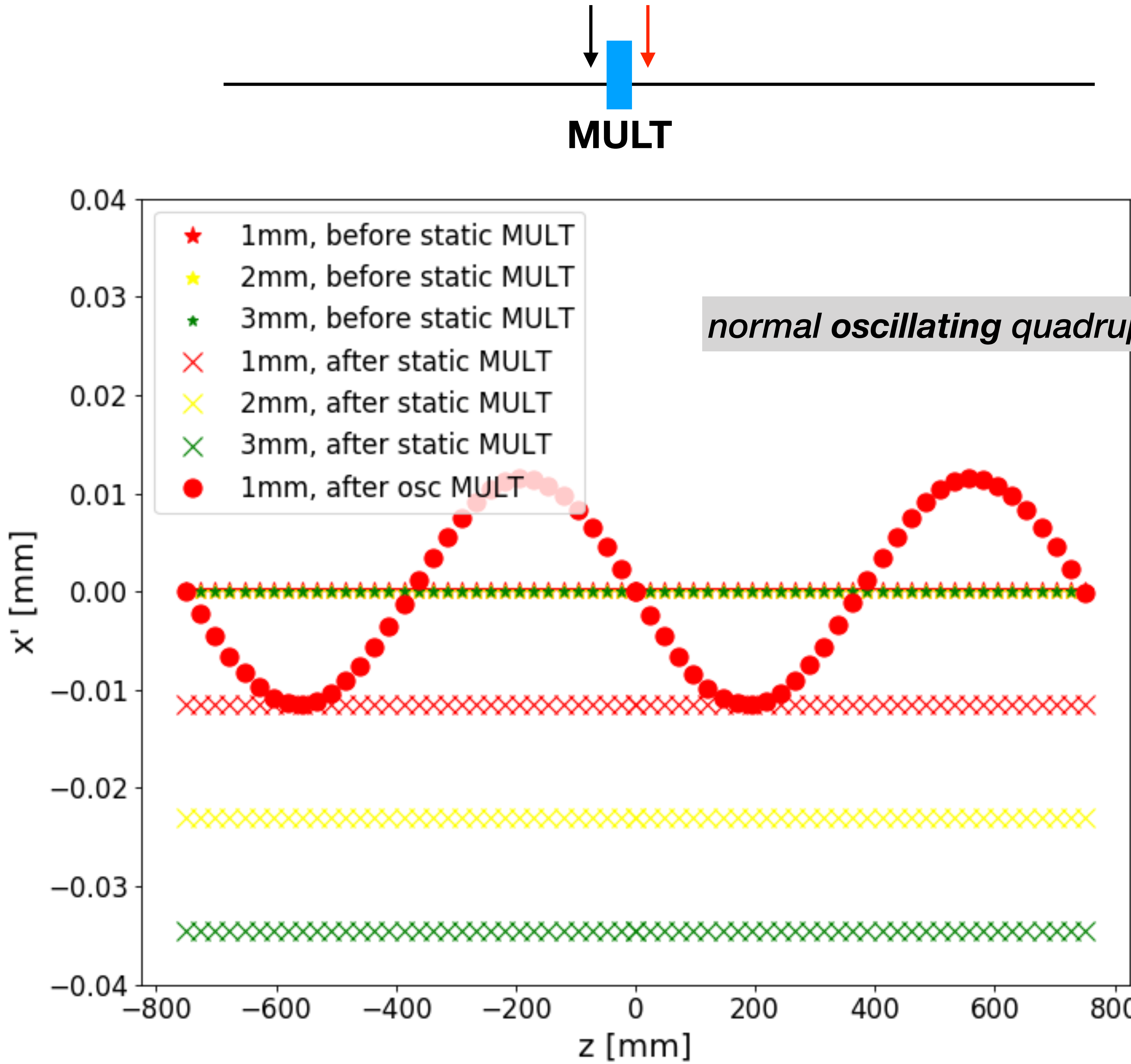

MULT
x=2 mm
x’=y=y’=0
normal static quadrupolar multipole (k1)
1mm, before static MULT
2mm, before static MULT
3mm, before static MULT
1mm, after static MULT
2mm, after static MULT
x = 1mm;
x’ = -0.0115307 mrad
x = 2mm;
x’ = -0.02306154 mrad
x' [mm]
z [mm]



MULT
x=3 mm
x’=y=y’=0
normal static quadrupolar multipole (k₁)
1mm, before static MULT
2mm, before static MULT
3mm, before static MULT
1mm, after static MULT
2mm, after static MULT
3mm, after static MULT
x = 1mm;
x’ = -0.0115307 mrad
x = 2mm;
x’ = -0.02306154 mrad
x = 3mm;
x’ = -0.03459231 mrad
x' [mm]
z [mm]
0.04
0.03
0.02
0.01
0.00
−0.01
−0.02
−0.03
−0.04
−800
−600
−400
−200
0
200
400
600
800

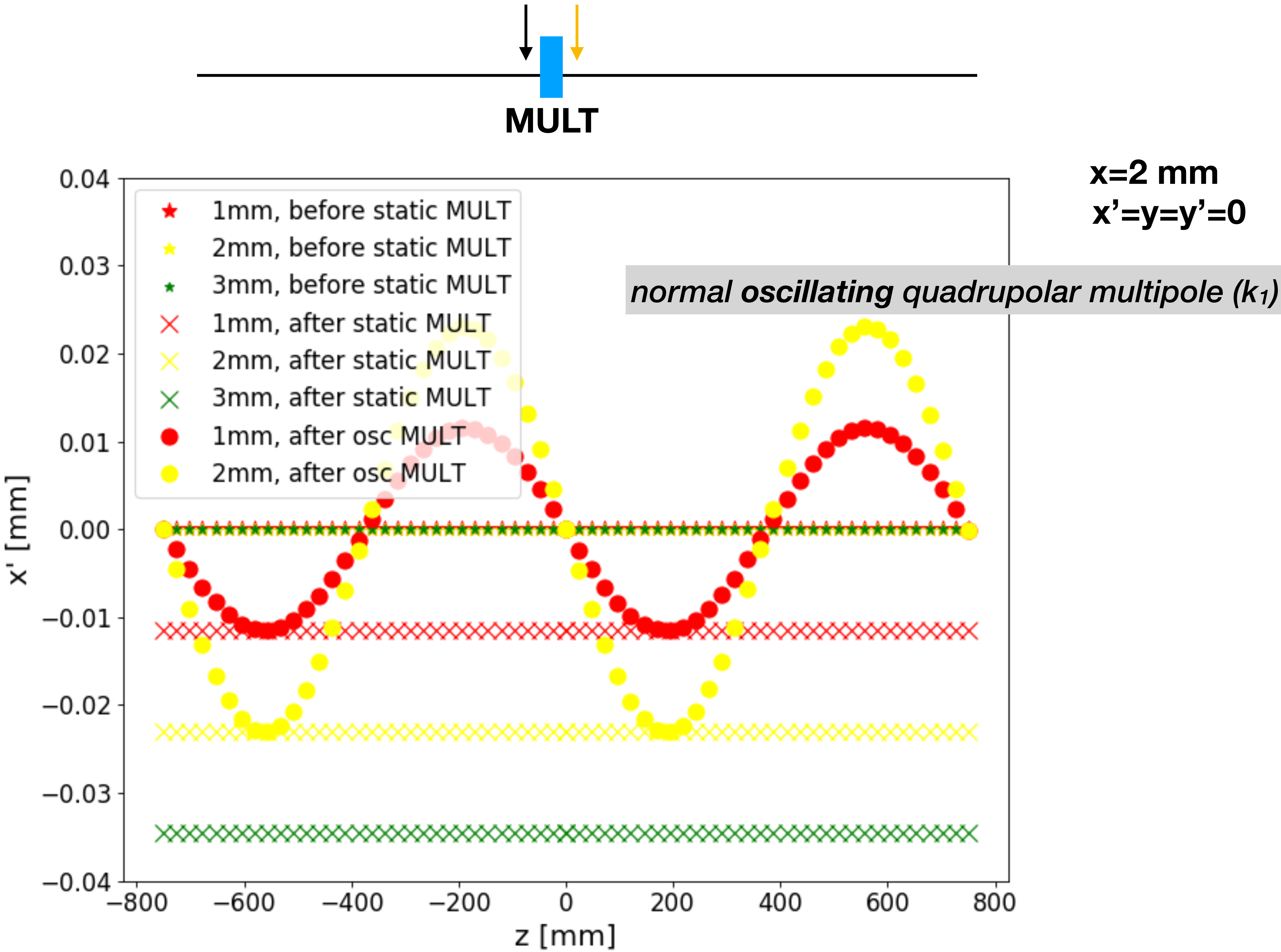


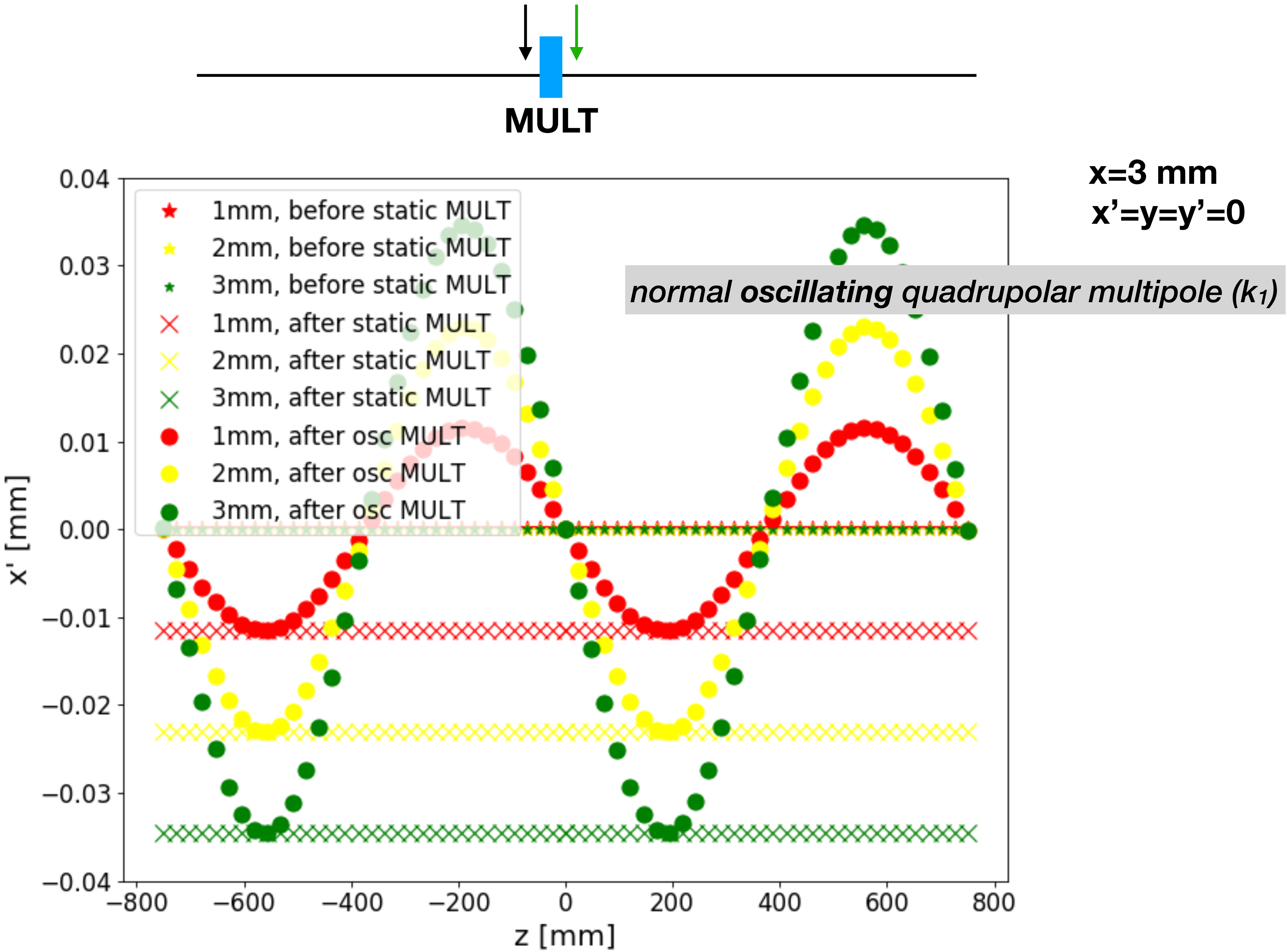
MULT
x=3 mm
x’=y=y’=0
normal static quadrupolar multipole (k₁)
linear relation of x and x’
1mm, before static MULT
2mm, before static MULT
3mm, before static MULT
1mm, after static MULT
2mm, after static MULT
3mm, after static MULT
x = 1mm;
x’ = -0.0115307 mrad
x = 2mm;
x’ = -0.02306154 mrad
x = 3mm;
x’ = -0.03459231 mrad
x' [mm]
z [mm]

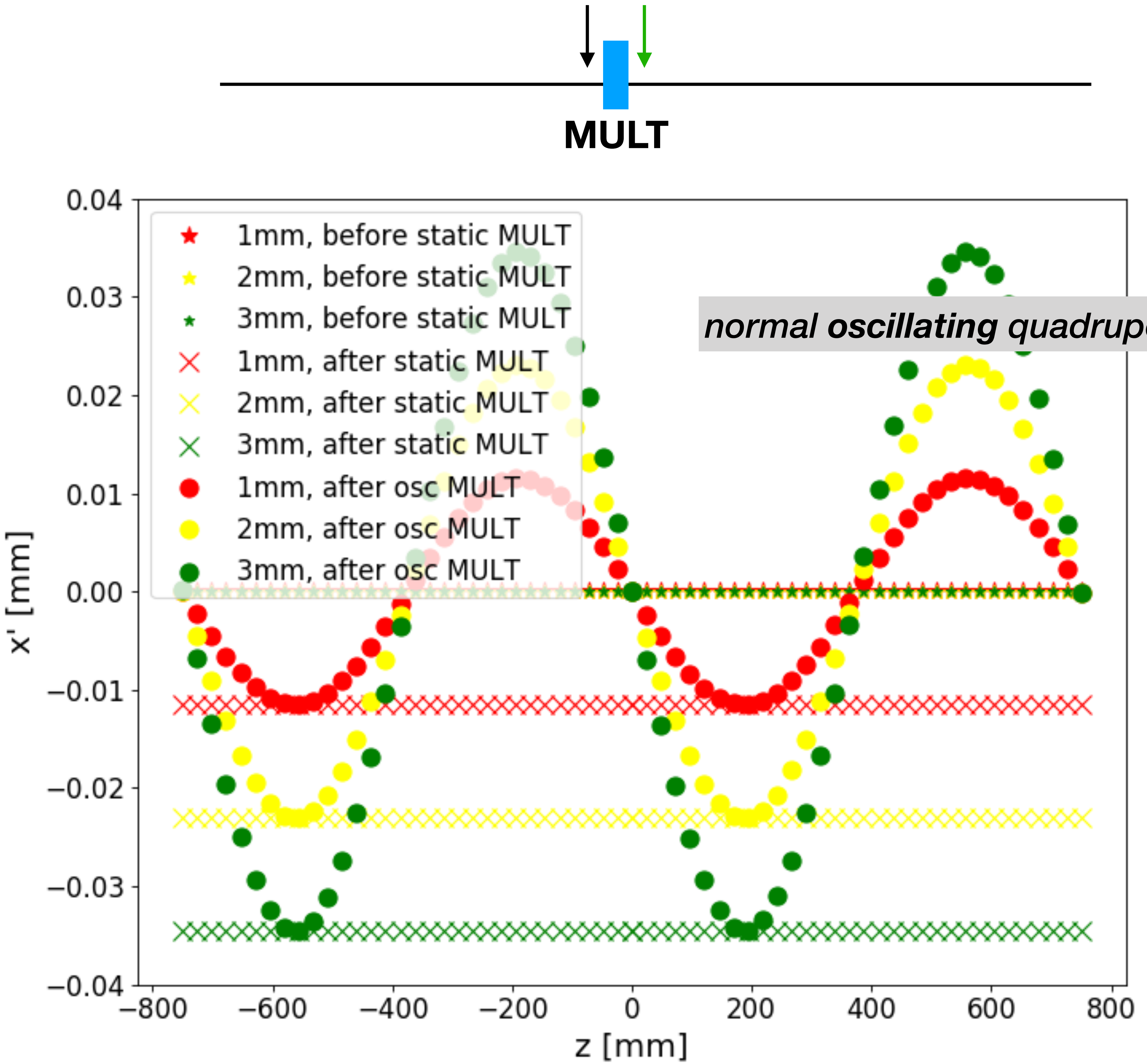


x=1 mm
x’=y=y’=0

normal oscillating quadrupolar multipole (k_1)

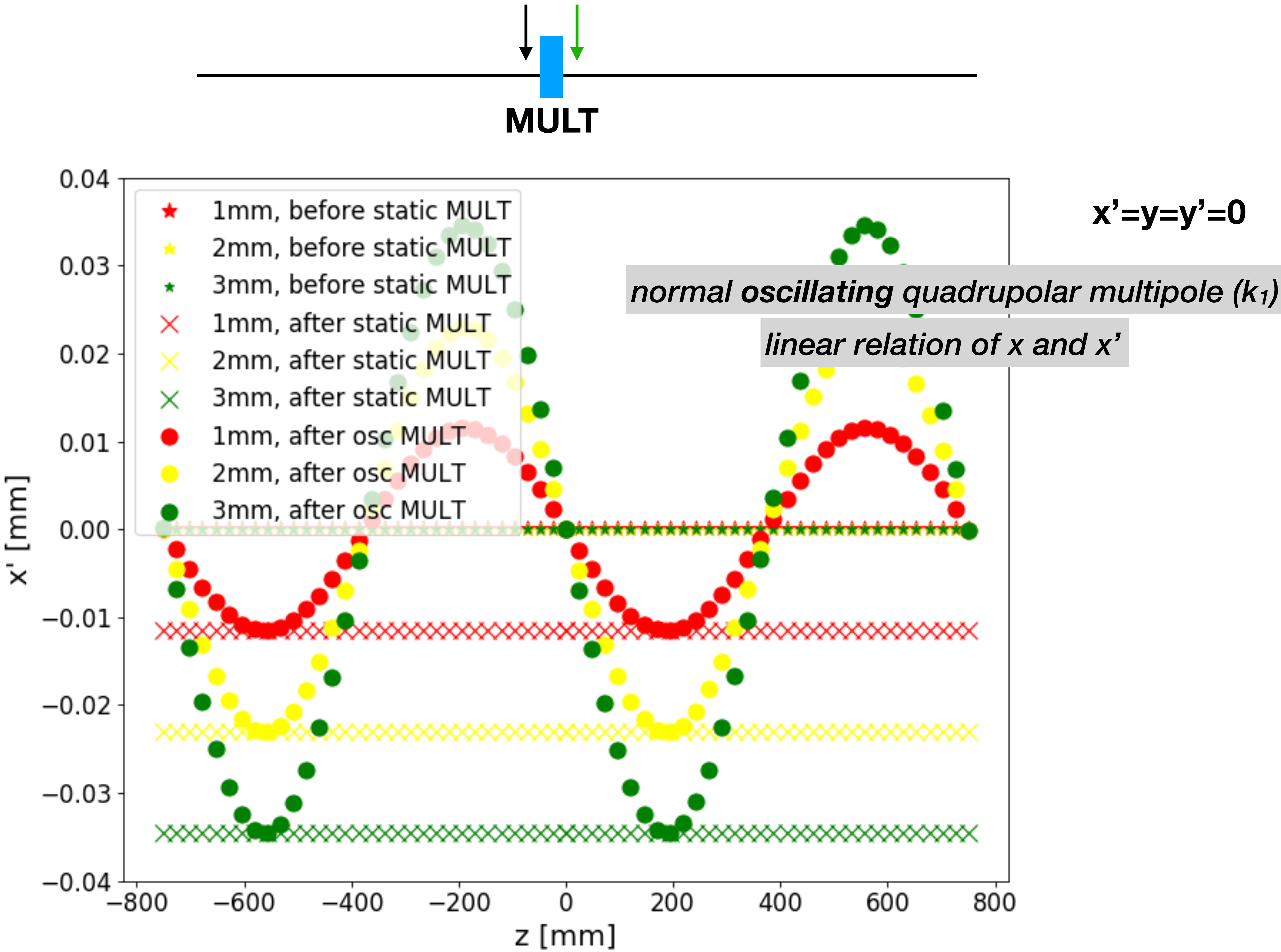

MULT
x=2 mm
x'=y=y'=0
normal oscillating quadrupolar multipole (k_1)
1mm, before static MULT
2mm, before static MULT
3mm, before static MULT
1mm, after static MULT
2mm, after static MULT
3mm, after static MULT
1mm, after osc MULT
2mm, after osc MULT
x' [mm]
z [mm]
0.04
0.03
0.02
0.01
0.00
−0.01
−0.02
−0.03
−0.04
−800
−600
−400
−200
0
200
400
600
800


MULT
x=3 mm
x'=y=y'=0
normal oscillating quadrupolar multipole (k1)
1mm, before static MULT
2mm, before static MULT
3mm, before static MULT
1mm, after static MULT
2mm, after static MULT
3mm, after static MULT
1mm, after osc MULT
2mm, after osc MULT
3mm, after osc MULT
x' [mm]
z [mm]
0.04
0.03
0.02
0.01
0.00
−0.01
−0.02
−0.03
−0.04
−800
−600
−400
−200
0
200
400
600
800



x’=y=y’=0

***normal oscillating** quadrupolar multipole (k_1)*


MULT
1mm, before static MULT
2mm, before static MULT
3mm, before static MULT
1mm, after static MULT
2mm, after static MULT
3mm, after static MULT
1mm, after osc MULT
2mm, after osc MULT
3mm, after osc MULT
x' [mm]
z [mm]
normal oscillating quadrupolar multipole (k1)
linear relation of x and x'

x'=y=y'=0

How to run SixTrack

How to run SixTrack

Files you need:

How to run SixTrack

Files you need:

a. mad-x script, e.g. “*madx_mask_file.madx*”: mask file that calls sequence and strengths; needs to have `makethin` and “`SIXTRACK, CAVALL, RADIUS = 17E-03;`”

How to run SixTrack

Files you need:

a. mad-x script, e.g. "*madx_mask_file.madx*": mask file that calls sequence and strengths; needs to have `makethin` and "`SIXTRACK, CAVALL, RADIUS = 17E-03;`"

b. *fort.13*: initial particle distribution* (see appendix)

*Disclaimer: fort.13 can be used to give initial distribution although that's not what it's intended for. The DIST block is intended for initial distribution (see "3.5 Initial Distribution from an ASCII file" of SixTrack-manual)

How to run SixTrack

Files you need:

a. mad-x script, e.g. "*madx_mask_file.madx*": mask file that calls sequence and strengths; needs to have `makethin` and "`SIXTRACK, CAVALL, RADIUS = 17E-03;`"

b. *fort.13*: initial particle distribution* (see appendix)

*Disclaimer: fort.13 can be used to give initial distribution although that's not what it's intended for. The DIST block is intended for initial distribution (see "3.5 Initial Distribution from an ASCII file" of SixTrack-manual)

How to run SixTrack

Files you need:

a. mad-x script, e.g. "*madx_mask_file.madx*": mask file that calls sequence and strengths; needs to have `makethin` and "`SIXTRACK, CAVALL, RADIUS = 17E-03;`"

b. *fort.13*: initial particle distribution* (see appendix)

c. fort.3: tracking and other parameters (see appendix)

*Disclaimer: fort.13 can be used to give initial distribution although that's not what it's intended for. The DIST block is intended for initial distribution (see "3.5 Initial Distribution from an ASCII file" of SixTrack-manual)

How to run SixTrack

Files you need:

a. mad-x script, e.g. “*madx_mask_file.madx*”: mask file that calls sequence and strengths; needs to have `makethin` and “`SIXTRACK, CAVALL, RADIUS = 17E-03;`”

b. *fort.13*: initial particle distribution* (see appendix)

c. fort.3: tracking and other parameters (see appendix)

1. Run mask file: `madx<madx_mask_file.madx`; this creates files *fc.2*, *fc.3.aux*, *fc.34*

*Disclaimer: fort.13 can be used to give initial distribution although that’s not what it’s intended for. The DIST block is intended for initial distribution (see “3.5 Initial Distribution from an ASCII file” of SixTrack-manual)

How to run SixTrack

Files you need:

a. mad-x script, e.g. "*madx_mask_file.madx*": mask file that calls sequence and strengths; needs to have `makethin` and "`SIXTRACK, CAVALL, RADIUS = 17E-03;`"

b. *fort.13*: initial particle distribution* (see appendix)

c. fort.3: tracking and other parameters (see appendix)

1. Run mask file: `madx<madx_mask_file.madx`; this creates files *fc.2*, *fc.3.aux*, *fc.34*

2. Rename created files: `mv fc.2 fort.2; mv fc.3.aux fort.3.aux; mv fc.34 fort.34;`

*Disclaimer: fort.13 can be used to give initial distribution although that's not what it's intended for. The DIST block is intended for initial distribution (see "3.5 Initial Distribution from an ASCII file" of SixTrack-manual)

How to run SixTrack

Files you need:

a. mad-x script, e.g. "*madx_mask_file.madx*": mask file that calls sequence and strengths; needs to have `makethin` and "`SIXTRACK, CAVALL, RADIUS = 17E-03;`"

b. *fort.13*: initial particle distribution* (see appendix)

c. fort.3: tracking and other parameters (see appendix)

1. Run mask file: `madx<madx_mask_file.madx`; this creates files *fc.2*, *fc.3.aux*, *fc.34*

2. Rename created files: `mv fc.2 fort.2; mv fc.3.aux fort.3.aux; mv fc.34 fort.34;`

3. In case you include machine errors, copy the content of *fc.3* in *fort.3*

*Disclaimer: fort.13 can be used to give initial distribution although that's not what it's intended for. The DIST block is intended for initial distribution (see "3.5 Initial Distribution from an ASCII file" of SixTrack-manual)

How to run SixTrack

Files you need:

a. mad-x script, e.g. "*madx_mask_file.madx*": mask file that calls sequence and strengths; needs to have `makethin` and "`SIXTRACK, CAVALL, RADIUS = 17E-03;`"

b. *fort.13*: initial particle distribution* (see appendix)

c. fort.3: tracking and other parameters (see appendix)

1. Run mask file: `madx<madx_mask_file.madx`; this creates files *fc.2*, *fc.3.aux*, *fc.34*
2. Rename created files: `mv fc.2 fort.2; mv fc.3.aux fort.3.aux; mv fc.34 fort.34;`
3. In case you include machine errors, copy the content of *fc.3* in *fort.3*
4. Run the SixTrack version you want: ./SixTrack_4619_crlibm_fast_tilt_cmake_Linux_gfortran_static_x86_64_64bit

*Disclaimer: fort.13 can be used to give initial distribution although that's not what it's intended for. The DIST block is intended for initial distribution (see "3.5 Initial Distribution from an ASCII file" of SixTrack-manual)

How to run SixTrack

Files you need:

a. mad-x script, e.g. "*madx_mask_file.madx*": mask file that calls sequence and strengths; needs to have `makethin` and "`SIXTRACK, CAVALL, RADIUS = 17E-03;`"

b. *fort.13*: initial particle distribution* (see appendix)

c. fort.3: tracking and other parameters (see appendix)

1. Run mask file: `madx<madx_mask_file.madx`; this creates files *fc.2*, *fc.3.aux*, *fc.34*
2. Rename created files: `mv fc.2 fort.2; mv fc.3.aux fort.3.aux; mv fc.34 fort.34;`
3. In case you include machine errors, copy the content of *fc.3* in *fort.3*
4. Run the SixTrack version you want: ./SixTrack_4619_crlibm_fast_tilt_cmake_Linux_gfortran_static_x86_64_64bit

Working example:
/afs/cern.ch/user/a/aalekou/public/bb_LumiMeeting_5Apr19/examples/install_CC

*Disclaimer: fort.13 can be used to give initial distribution although that's not what it's intended for. The DIST block is intended for initial distribution (see "3.5 Initial Distribution from an ASCII file" of SixTrack-manual)

DUMP the beam

From "9.1 Dumping of Beam Population" of <u>SixTrack-manual</u>

DUMP the beam

From "9.1 Dumping of Beam Population" of SixTrack-manual

- See your particle distribution at any location you want, just install a marker —> outputFile includes data on: nTurn, x, x', y, y', z, dpp

- Put following block in *fort.3*:

```
DUMP
ipmymarker.1 1 661 2 marker1-dump.txt
ipmymarker.2 1 662 2 anotherMarker-dump.txt
/ALL 1 665 2 elementByElement.txt
NEXT
```

DUMP the beam

From "9.1 Dumping of Beam Population" of SixTrack-manual

- See your particle distribution at any location you want, just install a marker —> outputFile includes data on: nTurn, x, x', y, y', z, dpp

- Put following block in *fort.3*:

marker's name

```
DUMP
ipmymarker.1 1 661 2 marker1-dump.txt
ipmymarker.2 1 662 2 anotherMarker-dump.txt
/ALL 1 665 2 elementByElement.txt
NEXT
```

DUMP the beam

From “9.1 Dumping of Beam Population” of SixTrack-manual

- See your particle distribution at any location you want, just install a marker —> outputFile includes data on: nTurn, x, x’, y, y’, z, dpp

- Put following block in *fort.3*:

marker’s name

output file-name

```
DUMP
ipmymarker.1 1 661 2 marker1-dump.txt
ipmymarker.2 1 662 2 anotherMarker-dump.txt
/ALL 1 665 2 elementByElement.txt
NEXT
```

DUMP the beam

From "9.1 Dumping of Beam Population" of SixTrack-manual

- See your particle distribution at any location you want, just install a marker —> outputFile includes data on: nTurn, x, x', y, y', z, dpp

- Put following block in *fort.3*:

marker's name

output file-name

```
DUMP
ipmymarker.1 1 661 2 marker1-dump.txt
ipmymarker.2 1 662 2 anotherMarker-dump.txt
/ALL 1 665 2 elementByElement.txt
NEXT
```

comment out if you want population dumping at each element

DYNK

From "5.5 Dynamic Kicks" of SixTrack-manual

- Increase voltage of cavity slowly to go to new CO that includes full CC kick in *fort.3*

```
DYNK
FUN crabVolt1 GET cravity.1 voltage
FUN crabVolt2 GET cravity.2 voltage
FUN ramp LIN 0.003333333333 0
FUN ramp_CC1 MUL crabVolt1 ramp
FUN ramp_CC2 MUL crabVolt2 ramp
SET cravity.1 voltage ramp_CC1 1 300 -1
SET cravity.2 voltage ramp_CC2 1 300 -1
NEXT
```

voltage and other output in *dynksets.dat*

DYNK

From "5.5 Dynamic Kicks" of SixTrack-manual

- Increase voltage of cavity slowly to go to new CO that includes full CC kick in *fort.3*

```
DYNK
FUN crabVolt1 GET cravity.1 voltage
FUN crabVolt2 GET cravity.2 voltage
FUN ramp LIN 0.003333333333 0
FUN ramp_CC1 MUL crabVolt1 ramp
FUN ramp_CC2 MUL crabVolt2 ramp
SET cravity.1 voltage ramp_CC1 1 300 -1
SET cravity.2 voltage ramp_CC2 1 300 -1
NEXT
```

voltage and other output in *dynksets.dat*

DYNK

From "5.5 Dynamic Kicks" of SixTrack-manual

- Increase voltage of cavity slowly to go to new CO that includes full CC kick in *fort.3*

FUN: function definition

```
DYNK
FUN crabVolt1 GET cravity.1 voltage
FUN crabVolt2 GET cravity.2 voltage
FUN ramp LIN 0.003333333333 0
FUN ramp_CC1 MUL crabVolt1 ramp
FUN ramp_CC2 MUL crabVolt2 ramp
SET cravity.1 voltage ramp_CC1 1 300 -1
SET cravity.2 voltage ramp_CC2 1 300 -1
NEXT
```

voltage and other output in *dynksets.dat*

DYNK

From "5.5 Dynamic Kicks" of SixTrack-manual

- Increase voltage of cavity slowly to go to new CO that includes full CC kick in *fort.3*

FUN: function definition

GET: get the original **voltage** (max_volt) of cravity.1 and name it "**crabVolt1**"

```
DYNK
FUN crabVolt1 GET cravity.1 voltage
FUN crabVolt2 GET cravity.2 voltage
FUN ramp LIN 0.003333333333 0
FUN ramp_CC1 MUL crabVolt1 ramp
FUN ramp_CC2 MUL crabVolt2 ramp
SET cravity.1 voltage ramp_CC1 1 300 -1
SET cravity.2 voltage ramp_CC2 1 300 -1
NEXT
```

voltage and other output in *dynksets.dat*

DYNK

From "5.5 Dynamic Kicks" of SixTrack-manual

- Increase voltage of cavity slowly to go to new CO that includes full CC kick in *fort.3*

`FUN`: function definition

`GET`: get the original **voltage** (max_volt) of cravity.1 and name it "**crabVolt1**"

```
DYNK
FUN crabVolt1 GET cravity.1 voltage
FUN crabVolt2 GET cravity.2 voltage
FUN ramp LIN 0.003333333333 0
FUN ramp_CC1 MUL crabVolt1 ramp
FUN ramp_CC2 MUL crabVolt2 ramp
SET cravity.1 voltage ramp_CC1 1 300 -1
SET cravity.2 voltage ramp_CC2 1 300 -1
NEXT
```

voltage and other output in *dynksets.dat*

DYNK

From “5.5 Dynamic Kicks” of SixTrack-manual

- Increase voltage of cavity slowly to go to new CO that includes full CC kick in *fort.3*

```
DYNK
FUN crabVolt1 GET cravity.1 voltage
FUN crabVolt2 GET cravity.2 voltage
FUN ramp LIN 0.003333333333 0
FUN ramp_CC1 MUL crabVolt1 ramp
FUN ramp_CC2 MUL crabVolt2 ramp
SET cravity.1 voltage ramp_CC1 1 300 -1
SET cravity.2 voltage ramp_CC2 1 300 -1
NEXT
```

FUN: function definition
GET: get the original **voltage** (max_volt) of cravity.1 and name it “**crabVolt1**”
LIN: computed value from:
y(t) = a*t + b = (1/300)*t (300: total number of turns)

voltage and other output in *dynksets.dat*

DYNK

From "5.5 Dynamic Kicks" of SixTrack-manual

- Increase voltage of cavity slowly to go to new CO that includes full CC kick in *fort.3*

```
DYNK
FUN crabVolt1 GET cravity.1 voltage
FUN crabVolt2 GET cravity.2 voltage
FUN ramp LIN 0.003333333333 0
FUN ramp_CC1 MUL crabVolt1 ramp
FUN ramp_CC2 MUL crabVolt2 ramp
SET cravity.1 voltage ramp_CC1 1 300 -1
SET cravity.2 voltage ramp_CC2 1 300 -1
NEXT
```

FUN: function definition

GET: get the original **voltage** (max_volt) of cravity.1 and name it "**crabVolt1**"

LIN: computed value from:

y(t) = a*t + b = (1/300)*t (300: total number of turns)

voltage and other output in *dynksets.dat*

DYNK

From "5.5 Dynamic Kicks" of SixTrack-manual

- Increase voltage of cavity slowly to go to new CO that includes full CC kick in *fort.3*

```
DYNK
FUN crabVolt1 GET cravity.1 voltage
FUN crabVolt2 GET cravity.2 voltage
FUN ramp LIN 0.003333333333 0
FUN ramp_CC1 MUL crabVolt1 ramp
FUN ramp_CC2 MUL crabVolt2 ramp
SET cravity.1 voltage ramp_CC1 1 300 -1
SET cravity.2 voltage ramp_CC2 1 300 -1
NEXT
```

FUN: function definition
GET: get the original **voltage** (max_volt) of cravity.1 and name it "**crabVolt1**"
LIN: computed value from:
y(t) = a*t + b = (1/300)*t (300: total number of turns)
MUL: multiply max_volt of cravity.1 with "**ramp**", i.e. (1/300)*t and put value in "**ramp_CC1**"

voltage and other output in *dynksets.dat*

DYNK

From "5.5 Dynamic Kicks" of SixTrack-manual

- Increase voltage of cavity slowly to go to new CO that includes full CC kick in *fort.3*

```
DYNK
FUN crabVolt1 GET cravity.1 voltage
FUN crabVolt2 GET cravity.2 voltage
FUN ramp LIN 0.003333333333 0
FUN ramp_CC1 MUL crabVolt1 ramp
FUN ramp_CC2 MUL crabVolt2 ramp
SET cravity.1 voltage ramp_CC1 1 300 -1
SET cravity.2 voltage ramp_CC2 1 300 -1
NEXT
```

FUN: function definition
GET: get the original **voltage** (max_volt) of cravity.1 and name it "**crabVolt1**"
LIN: computed value from:
y(t) = a*t + b = (1/300)*t (300: total number of turns)
MUL: multiply max_volt of cravity.1 with "**ramp**", i.e. (1/300)*t and put value in "**ramp_CC1**"

voltage and other output in *dynksets.dat*

DYNK

From "5.5 Dynamic Kicks" of SixTrack-manual

- Increase voltage of cavity slowly to go to new CO that includes full CC kick in *fort.3*

```
DYNK
FUN crabVolt1 GET cravity.1 voltage
FUN crabVolt2 GET cravity.2 voltage
FUN ramp LIN 0.003333333333 0
FUN ramp_CC1 MUL crabVolt1 ramp
FUN ramp_CC2 MUL crabVolt2 ramp
SET cravity.1 voltage ramp_CC1 1 300 -1
SET cravity.2 voltage ramp_CC2 1 300 -1
NEXT
```

FUN: function definition

GET: get the original **voltage** (max_volt) of cravity.1 and name it "**crabVolt1**"

LIN: computed value from:
y(t) = a*t + b = (1/300)*t (300: total number of turns)

MUL: multiply max_volt of cravity.1 with "**ramp**", i.e. (1/300)*t and put value in "**ramp_CC1**"

SET: set the voltage of cravity.1 to "**ramp_CC1**", do this from t=1 to t=300

voltage and other output in *dynksets.dat*

DYNK

From "5.5 Dynamic Kicks" of SixTrack-manual

- Increase voltage of cavity slowly to go to new CO that includes full CC kick in *fort.3*

```
DYNK
FUN crabVolt1 GET cravity.1 voltage
FUN crabVolt2 GET cravity.2 voltage
FUN ramp LIN 0.003333333333 0
FUN ramp_CC1 MUL crabVolt1 ramp
FUN ramp_CC2 MUL crabVolt2 ramp
SET cravity.1 voltage ramp_CC1 1 300 -1
SET cravity.2 voltage ramp_CC2 1 300 -1
NEXT
```

`FUN`: function definition
`GET`: get the original **voltage** (max_volt) of cravity.1 and name it "**crabVolt1**"
`LIN`: computed value from:
y(t) = a*t + b = (1/300)*t (300: total number of turns)
`MUL`: multiply max_volt of cravity.1 with "**ramp**", i.e. (1/300)*t and put value in "**ramp_CC1**"
`SET`: set the voltage of cravity.1 to "**ramp_CC1**", do this from t=1 to t=300

voltage and other output in *dynksets.dat*

DYNK

From “5.5 Dynamic Kicks” of SixTrack-manual

- Increase voltage of cavity slowly to go to new CO that includes full CC kick in *fort.3*

```
DYNK
FUN crabVolt1 GET cravity.1 voltage
FUN crabVolt2 GET cravity.2 voltage
FUN ramp LIN 0.003333333333 0
FUN ramp_CC1 MUL crabVolt1 ramp
FUN ramp_CC2 MUL crabVolt2 ramp
SET cravity.1 voltage ramp_CC1 1 300 -1
SET cravity.2 voltage ramp_CC2 1 300 -1
NEXT
```

voltage and other output in *dynksets.dat*

FUN: function definition
GET: get the original **voltage** (max_volt) of cravity.1 and name it “**crabVolt1**”
LIN: computed value from:
y(t) = a*t + b = (1/300)*t (300: total number of turns)
MUL: multiply max_volt of cravity.1 with “**ramp**”, i.e. (1/300)*t and put value in “**ramp_CC1**”
SET: set the voltage of cravity.1 to “**ramp_CC1**”, do this from t=1 to t=300

So in last turn, t=300, voltage of cravity.1 is **SET** to ramp_CC1

DYNK

From "5.5 Dynamic Kicks" of SixTrack-manual

- Increase voltage of cavity slowly to go to new CO that includes full CC kick in *fort.3*

```
DYNK
FUN crabVolt1 GET cravity.1 voltage
FUN crabVolt2 GET cravity.2 voltage
FUN ramp LIN 0.003333333333 0
FUN ramp_CC1 MUL crabVolt1 ramp
FUN ramp_CC2 MUL crabVolt2 ramp
SET cravity.1 voltage ramp_CC1 1 300 -1
SET cravity.2 voltage ramp_CC2 1 300 -1
NEXT
```

voltage and other output in *dynksets.dat*

`FUN`: function definition

`GET`: get the original **voltage** (max_volt) of cravity.1 and name it "**crabVolt1**"

`LIN`: computed value from:
y(t) = a*t + b = (1/300)*t (300: total number of turns)

`MUL`: multiply max_volt of cravity.1 with "**ramp**", i.e. (1/300)*t and put value in "**ramp_CC1**"

`SET`: set the voltage of cravity.1 to "**ramp_CC1**", do this from t=1 to t=300

So in last turn, t=300, voltage of cravity.1 is **`SET`** to ramp_CC1 =max_volt*(1/300)*300

DYNK

From "5.5 Dynamic Kicks" of SixTrack-manual

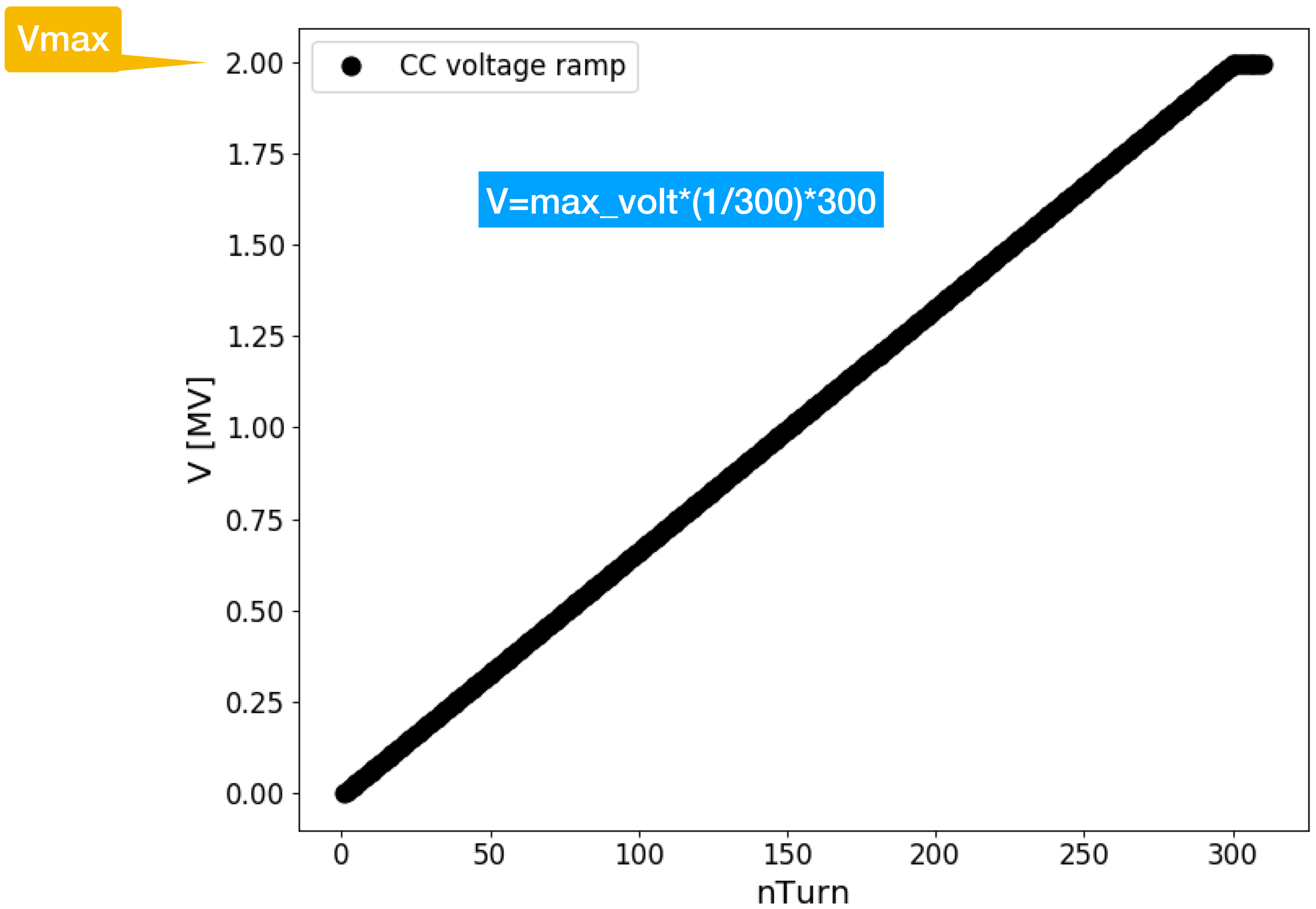
- Increase voltage of cavity slowly to go to new CO that includes full CC kick in *fort.3*

```
DYNK
FUN crabVolt1 GET cravity.1 voltage
FUN crabVolt2 GET cravity.2 voltage
FUN ramp LIN 0.003333333333 0
FUN ramp_CC1 MUL crabVolt1 ramp
FUN ramp_CC2 MUL crabVolt2 ramp
SET cravity.1 voltage ramp_CC1 1 300 -1
SET cravity.2 voltage ramp_CC2 1 300 -1
NEXT
```

voltage and other output in *dynksets.dat*

FUN: function definition
GET: get the original **voltage** (max_volt) of cravity.1 and name it "**crabVolt1**"
LIN: computed value from:
y(t) = a*t + b = (1/300)*t (300: total number of turns)
MUL: multiply max_volt of cravity.1 with "**ramp**", i.e. (1/300)*t and put value in "**ramp_CC1**"
SET: set the voltage of cravity.1 to "**ramp_CC1**", do this from t=1 to t=300

So in last turn, t=300, voltage of cravity.1 is **SET** to ramp_CC1
=max_volt*(1/300)*300
=max_volt



Vmax
CC voltage ramp
V=max_volt*(1/300)*300
V [MV]
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
50
100
150
200
250
300
nTurn

SixDesk

SixDesk

- Manual

SixDesk

- Manual
- “[…] run a tracking campaign, on either the CERN LSF batch system or BOINC, using the **familiar SixTrack run environment** on Linux […]”

SixDesk

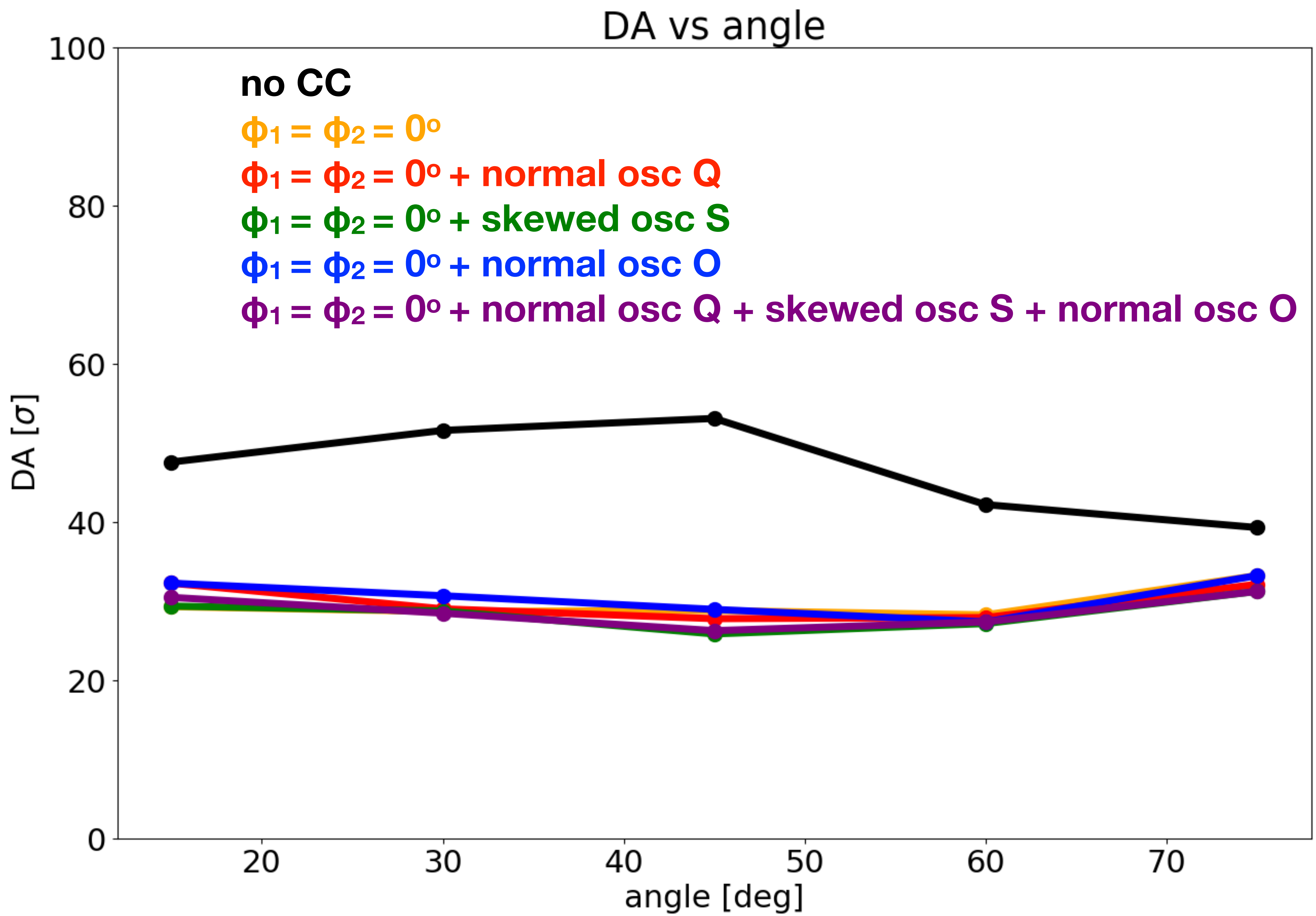
- Manual

- "[…] run a tracking campaign, on either the CERN LSF batch system or BOINC, using the **familiar SixTrack run environment** on Linux […]"

- Essential to first do small checks and confirm your beam behaves as expected **before** using SixDesk for DA runs

SixTrack/SixDesk related studies done to date for SPS

1. Study of RF multipoles effect on DA with angle
2. Study of DA_{min} for different skew sextupolar CC values
3. Study of DA for different V_{CC} and $z_{initial}$ in the presence of SPS multipole errors
4. FMA studies using pyNaff [F. Asvesta et al.]
5. Study of emittance increase from power supply ripple [N. Triantafyllou et al., 2nd part of this presentation]


DA vs angle
no CC
ϕ1 = ϕ2 = 0o
ϕ1 = ϕ2 = 0o + normal osc Q
ϕ1 = ϕ2 = 0o + skewed osc S
ϕ1 = ϕ2 = 0o + normal osc O
ϕ1 = ϕ2 = 0o + normal osc Q + skewed osc S + normal osc O
DA [σ]
angle [deg]
0
20
40
60
80
100
20
30
40
50
60
70

SixTrack/SixDesk related studies done to date for SPS

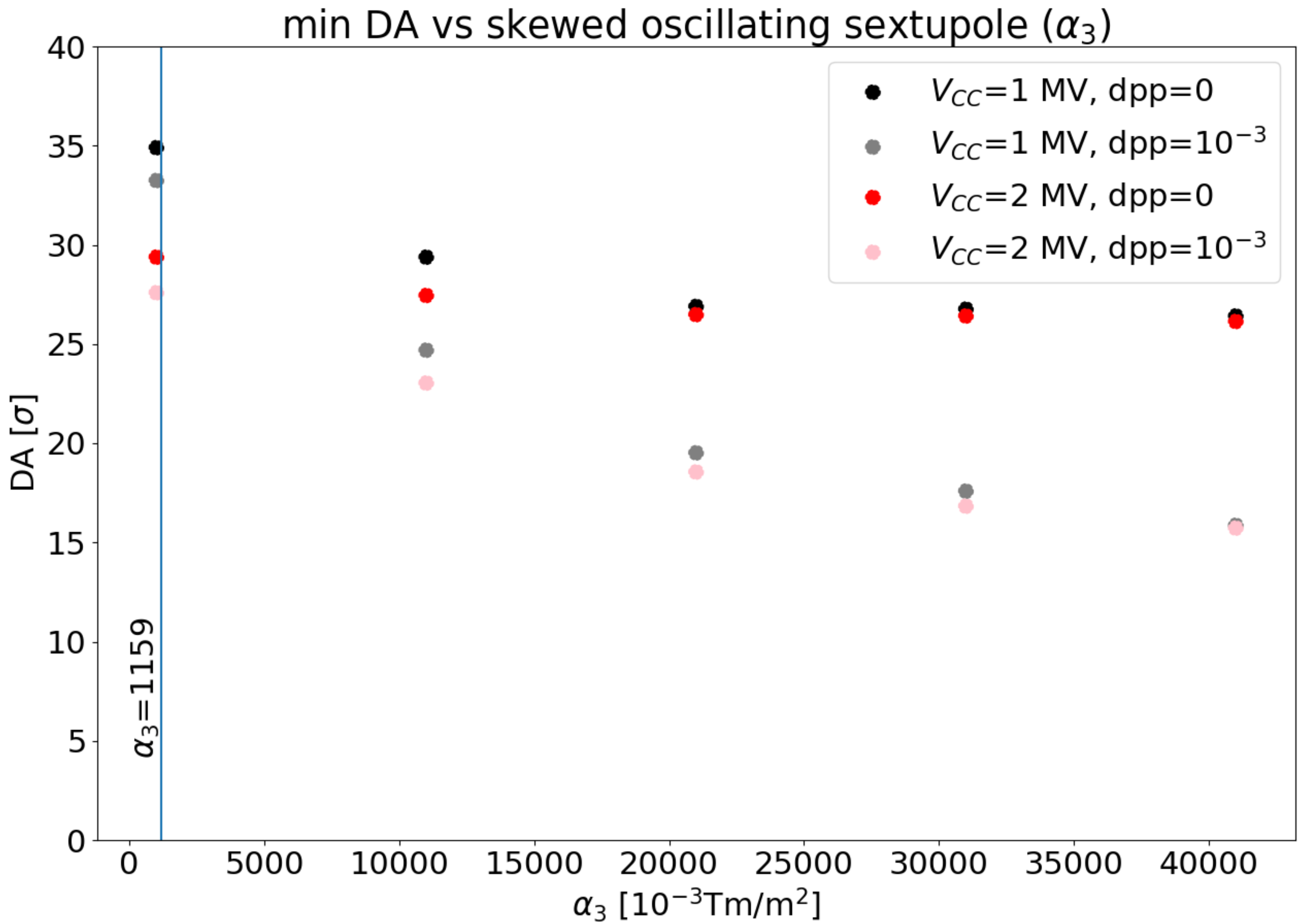
1. Study of RF multipoles effect on DA with angle

2. Study of DA_{min} for different skew sextupolar CC values

3. Study of DA for different V_{CC} and $z_{initial}$ in the presence of SPS multipole errors

4. FMA studies using pyNaff [F. Asvesta et al.]

5. Study of emittance increase from power supply ripple [N. Triantafyllou et al., 2nd part of this presentation]



min DA vs skewed oscillating sextupole (α_3)
V_{CC}=1 MV, dpp=0
V_{CC}=1 MV, dpp=10^{-3}
V_{CC}=2 MV, dpp=0
V_{CC}=2 MV, dpp=10^{-3}
α_3=1159
DA [σ]
α_3 [10^{-3}Tm/m^2]
0
5
10
15
20
25
30
35
40
0
5000
10000
15000
20000
25000
30000
35000
40000

SixTrack/SixDesk related studies done to date for SPS

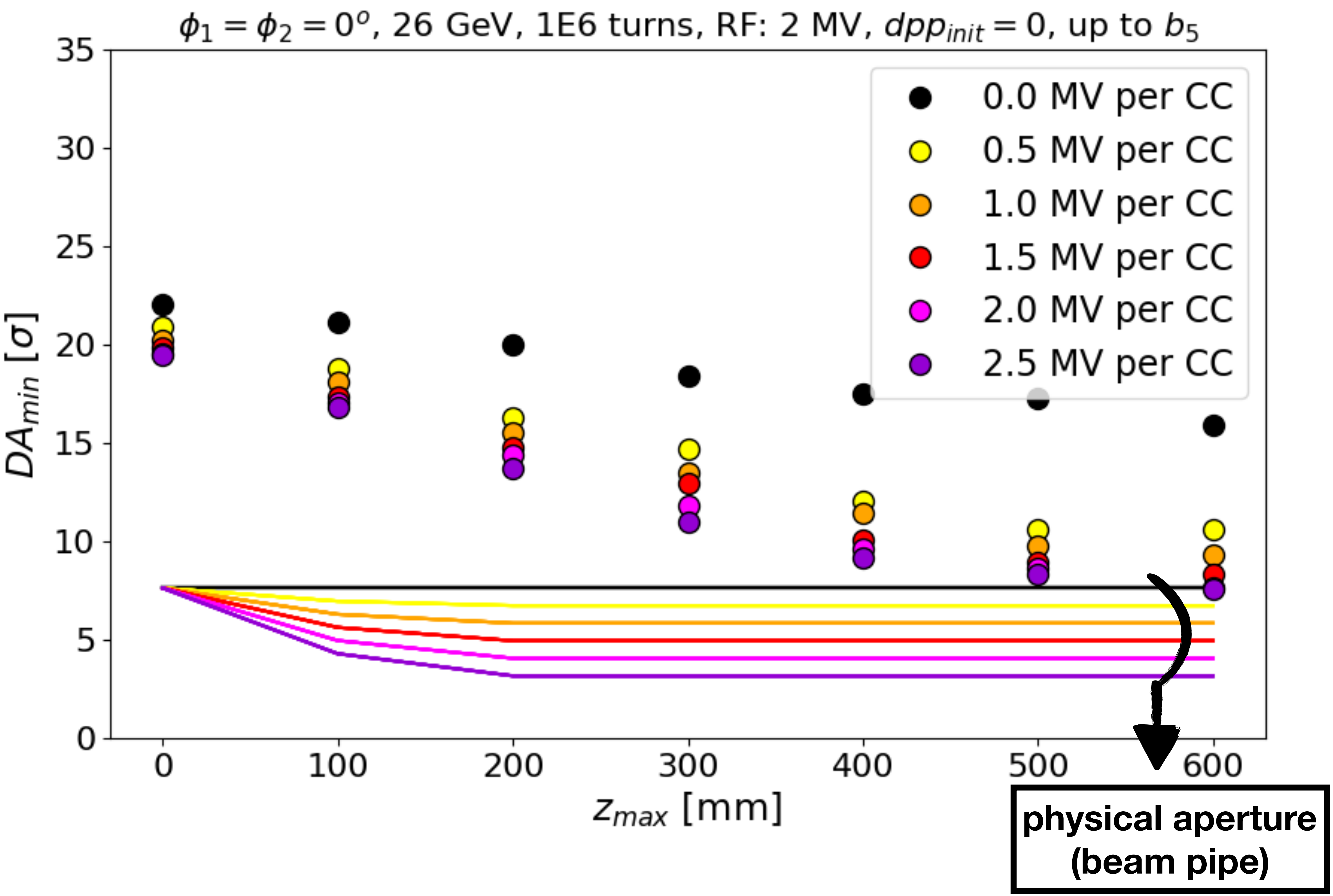
1. Study of RF multipoles effect on DA with angle

2. Study of DA_{min} for different skew sextupolar CC values

3. Study of DA for different V_{CC} and $z_{initial}$ in the presence of SPS multipole errors

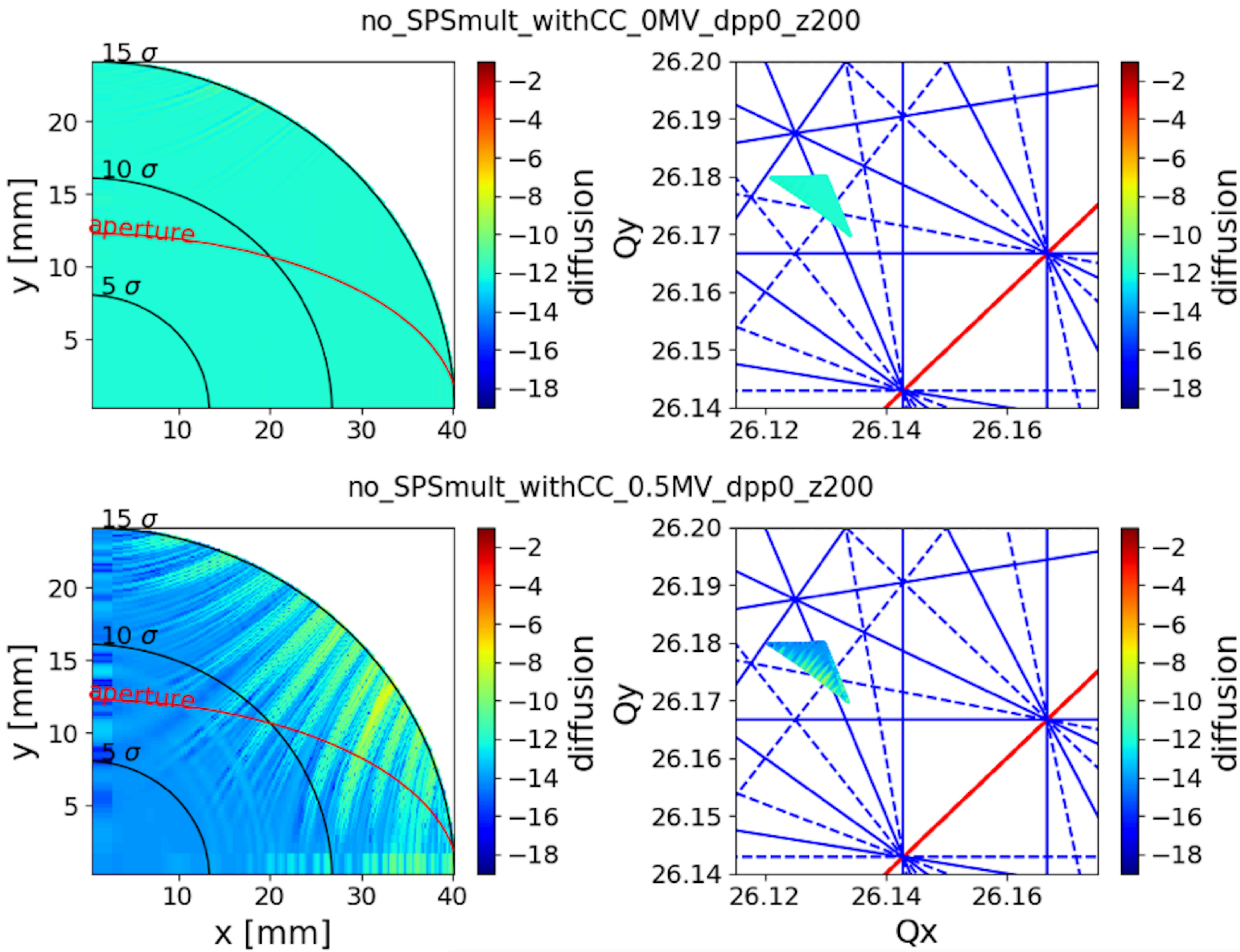
4. FMA studies using pyNaff [F. Asvesta et al.]

5. Study of emittance increase from power supply ripple [N. Triantafyllou et al., 2nd part of this presentation]


$\phi_1 = \phi_2 = 0^o$, 26 GeV, 1E6 turns, RF: 2 MV, $dpp_{init} = 0$, up to b_5
0
5
10
15
20
25
30
35
DA_{min} [σ]
0
100
200
300
400
500
600
z_{max} [mm]
0.0 MV per CC
0.5 MV per CC
1.0 MV per CC
1.5 MV per CC
2.0 MV per CC
2.5 MV per CC
physical aperture
(beam pipe)

SixTrack/SixDesk related studies done to date for SPS

1. Study of RF multipoles effect on DA with angle
2. Study of DA_{min} for different skew sextupolar CC values
3. Study of DA for different V_{CC} and $z_{initial}$ in the presence of SPS multipole errors
4. FMA studies using pyNaff [F. Asvesta et al.]
5. Study of emittance increase from power supply ripple [N. Triantafyllou et al., 2nd part of this presentation]



no_SPSmult_withCC_0MV_dpp0_z200
no_SPSmult_withCC_0.5MV_dpp0_z200
15 σ
10 σ
5 σ
aperture
y [mm]
x [mm]
diffusion
Qy
Qx

SixTrack/SixDesk related studies done to date for SPS

1. Study of RF multipoles effect on DA with angle

2. Study of DA_{min} for different skew sextupolar CC values

3. Study of DA for different V_{CC} and $z_{initial}$ in the presence of SPS multipole errors

4. FMA studies using pyNaff [F. Asvesta et al.]

5. Study of emittance increase from power supply ripple [N. Triantafyllou et al., 2nd part of this presentation]

Appendix

fort.3

```
GEOM
PRINTOUT
NEXT
TRAC
100 0 32 0 0 0 1
1 1 2 1 2
0 0 1 1 1 20000 2
NEXT
INIT
2 0 0 1
0.0
0.0
0.0
0.0
0.0
2e-3
0.0
0.0
0.0
0.0
0.0
2e-3
55000.0
55110.00851227352
55110.00851227352
NEXT
ITER
50 1.0E-12 1.0E-15
10 1.0E-10 1.0E-10
10 1.0E-10 1.0E-10
1.0E-09 1.0E-09 1.0E-09
NEXT
LINE
ELEMENT  0 1 1 2.5 2.5
NEXT
BEAM
2.2E+11 3.5 3.5 0.0755 1.13E-04 1 1 1 1
NEXT
SYNC
4620 0.001908372003 5. 0. 6911.503800 938.2796 1
1 1
NEXT
LINE
ELEMENT 0 2 1 2.5 2.5
NEXT
ENDE
```

TRACKING
number of turns, number of pairs…

INITIAL DISTRIBUTION
Important note: even if you give an external distribution (e.g. with fort.13), make sure you have the correct energies in fort.3, as CO is calculated here first, before reading the external distribution

BEAM
Beam-beam effect, nParticles, emittance, …

SYNCHROTRON OSCILLATION
harmonic number, compaction factor, RF voltage, circumference, …

Only first 4 letters of each block are being read e.g. “TRACKING” —> “TRAC” etc

fort.13

From “3.2 Initial Coordinates” of SixTrack-manual

0.0	x	of 1st particle
0.0	x’	
0.0	y	
0.0	y'	
-0.0	z	
0.0	dpp	
0.0	x	of 2nd particle
0.0	x’	
0.0	y	
0.0	y’	
0.0	z	
0.0	dpp	
26000.0	E0	**Energy of reference particle**
26000.0	E1	**Energy of 1st particle**
26000.0	E2	**Energy of 2nd particle**

Disclaimer: fort.13 can be used to give initial distribution although that’s not what it’s intended for. The DIST block is intended for initial distribution (see “3.5 Initial Distribution from an ASCII file” of SixTrack-manual)

fort.2

- GO flag: element-location where initial distribution starts
- In *fort.2*, under `STRUCTURE INPUT`, find the relevant location (e.g. cravity.1) and put a GO flag just before, i.e. “`GO cravity.1`”

Any questions?

- Just come find me! :)
- Very Important and Helpful People: Riccardo de Maria, A. Mereghetti and V. Olsen (MAD-X, SixTrack and SixDesk)