Software for the beam-based alignment of a RF-Photogun

Chris Marvin Zibula

B-TU Cottbus-Senftenberg

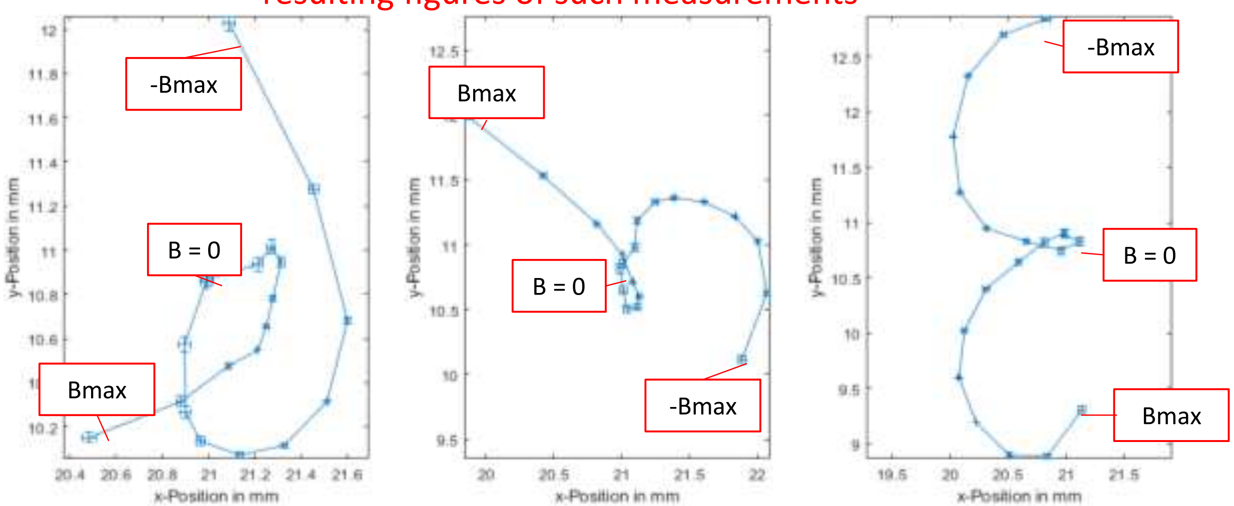
Brandenburgische Technische Universität Cottbus - Senftenberg

DESY.

Motivation

- To accelerate the electrons ideally the laser pulse position on the cathode and the solenoid are aligned
- To approach a perfect alignment, misalignments must be quantified and corrected for. A method is proposed using the positions of the electrons on the YAG screen of the photogun, measured as a function of the magnetic field strength B in the solenoid

resulting figures of such measurements



- The tighter the points to each other, the better the alignment
- Based on those measurements, misalignment can be simulated

Electromagnetic Fields of the Photogun

- Electric Field inside the Gun-Cavity is described by:

$$E_z(z,t) = E_0 \cdot E_{z,norm} \cdot \sin(\omega t + \varphi_0)$$

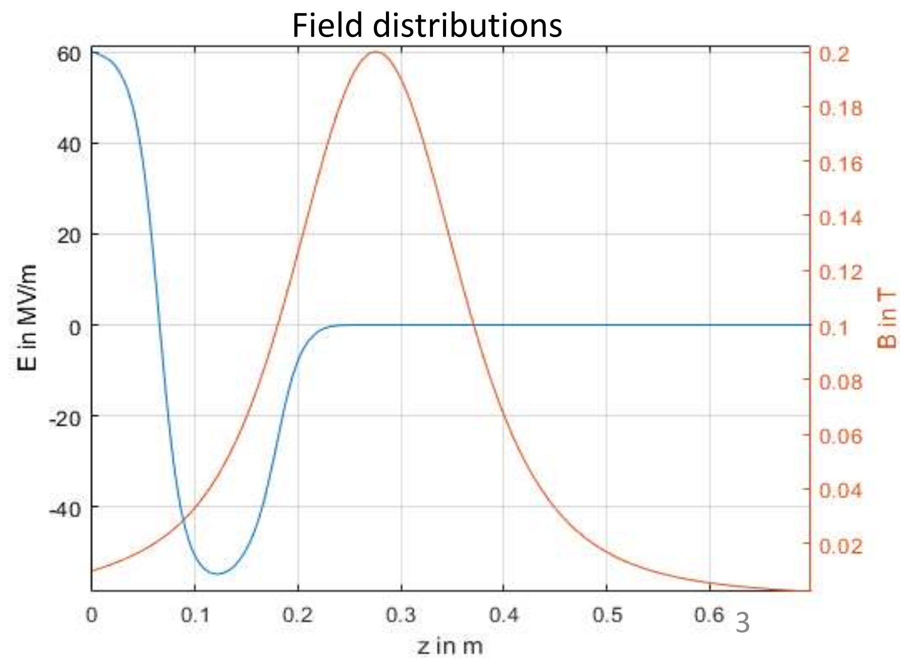
E_0 - Amplitude of the electric field $\sim 60\ {}^{MV}/_{m}$
$E_{z,norm}$ - Normalized field distribution of the cathode
ω - Angular frequency of the standing wave
φ_0 - Phase of the wave at release of electrons

- Magnetic Field of the Solenoid is described by:

$$B_z(z) = B_{max} \cdot B_{z,norm}(z)$$

B_{max} - maximum field strength $\sim 0.2\ T$
$B_{z,norm}$ - for maximum normalized field distribution

- With z being the axis along the electrons motion



Equations of motion

- The force equation of the motion of an electron:

$$\frac{d\vec{P}}{dt} = \overrightarrow{F_L} = \overrightarrow{F_E} + \overrightarrow{F_B}$$

- The velocity:

$$\frac{d\vec{x}}{dt} = \vec{v}$$

- With:

t	- time
$\vec{P} = \gamma m \vec{v}$	- momentum
$\overrightarrow{F_L}$	- Lorentz force
$\overrightarrow{F_E} = e\vec{E}$	- force by the electric field
$\overrightarrow{F_B} = e\vec{v} \times \vec{B}$	- force by the magnetic field
$\vec{x}$	- position
$\vec{v}$	- velocity
$\gamma = \frac{1}{\sqrt{1 - v^2/c^2}}$	- Lorentz factor

Equations of motion

- Using $\vec{\beta} = \frac{\vec{v}}{c}$ yields:

$$\frac{d}{dt}\frac{m\vec{v}}{\sqrt{1-\frac{v^2}{c^2}}} = e \cdot \vec{E} + e\vec{v} \times \vec{B}$$

$$\frac{d}{dt}\gamma\vec{\beta} = \frac{e}{mc}\vec{E} + \frac{e}{m}\vec{\beta} \times \vec{B}$$

- That results into the 6-dimensional differential equation system:

$$\frac{d}{d\tau}\vec{p} = \frac{e}{m\omega c}\vec{E} + \frac{e}{m\omega}\frac{\vec{p}}{\sqrt{1+p^2}} \times \vec{B}$$

$$\frac{d}{d\tau}\vec{\chi} = \frac{\vec{p}}{\sqrt{1+p^2}}$$

- With: $\tau = \omega t$ - dimensionless time

  $\vec{\beta} = \frac{v}{c} = \frac{\vec{p}}{\sqrt{1+p^2}}$ - dimensionless velocity

  $\vec{p} = \vec{\beta}\gamma$ - dimensionless momentum

  $\vec{\chi} = \vec{x}\frac{\omega}{c}$ - dimensionless position

Numerical solution

- 4^{th} order Runge-Kutta algorithm
- Time is discretized into N intervals per period
  → every time step is $\Delta\tau = \frac{2\pi}{N}$ long
- The system of differential equation is defined as $\vec{F}$ and its solution as $\vec{Y}$ such as:

$$\vec{Y}(\tau) = \begin{pmatrix} \vec{p} \\ \vec{\chi} \end{pmatrix}$$

$$\frac{d}{d\tau}\vec{Y}(\tau) = \vec{F}\left(\tau, \vec{Y}\right) = \begin{pmatrix} \frac{e}{m\omega c}\vec{E}\left(\tau, \vec{Y}\right) + \frac{e}{m\omega}\frac{\vec{p}}{\sqrt{1+p^2}} \times \vec{B}\left(\tau, \vec{Y}\right) \\ \frac{\vec{p}}{\sqrt{1+p^2}} \end{pmatrix}$$

Numerical solution

- With the starting conditions

$$\vec{Y}(\tau = 0) = \begin{pmatrix} \overrightarrow{p_0} \\ \overrightarrow{\chi_0} \end{pmatrix}, \qquad \text{with } \overrightarrow{p_0} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \text{ and } \overrightarrow{\chi_0} = \begin{pmatrix} \chi_{x,0} \\ \chi_{y,0} \\ 0 \end{pmatrix}$$

- χ_0 is where the laser hits the cathode
- For the (n+1)-st step there are the following temporary solutions:

$$\vec{K}_{n,1} = \Delta\tau \cdot \vec{F}(\tau_n, \vec{Y}_n)$$

$$\vec{K}_{n,2} = \Delta\tau \cdot \vec{F}\left(\tau_n + \tfrac{\Delta\tau}{2}, \vec{Y}_n + \tfrac{\vec{K}_{n,1}}{2}\right)$$

$$\vec{K}_{n,3} = \Delta\tau \cdot \vec{F}\left(\tau_n + \tfrac{\Delta\tau}{2}, \vec{Y}_n + \tfrac{\vec{K}_{n,2}}{2}\right)$$

$$\vec{K}_{n,4} = \Delta\tau \cdot \vec{F}\left(\tau_n + \Delta\tau, \vec{Y}_n + \vec{K}_{n,4}\right)$$

- Which will be added to the previous solution $\vec{Y}_n$ as follows:

$$\vec{Y}_{n+1} = \vec{Y}_n + \frac{\overrightarrow{K_{n,1}}}{6} + \frac{\overrightarrow{K_{n,2}}}{3} + \frac{\overrightarrow{K_{n,3}}}{3} + \frac{\overrightarrow{K_{n,4}}}{6}$$

Coding of the simulation

- The simulation is coded with MATLAB
- Three classes have been made:
  - The fields of the RF-Gun
  - The magnetic field of the solenoid
  - The tracker which is simulating the path of one electron
- The tracker uses one object of each class to calculate the momentum and location of the electron

Class of the RF-Gun

```
classdef RF_Field < handle
    properties
        filename        % filename of field distribution
        c = 299792458   % lightspeed m/s
        f = 1.3e9       % 1.3GHz
        k               % wavenumber
        z               % z-values of field distribution
        zeta            % dim-less z-values
        phi0            % startingphase
        E0              % amplitude of the electric field
        Ez_norm         % normed field distribution
        Ez_1dif         % 1st differentiation of normed field
        Ez_2dif         % 2nd diff
        Ez_3dif         % 3rd diff
    end
    methods
        function obj = RF_Field(E_filename, E_0)    % constructor ...

        function obj = setE0(obj, E_0)              % setting amplitude ...

        function obj = setPhi0(obj, phi0_degrees)   % set starting phase ...

        function deriv = GetDeriv(obj,Fz,z)         % derivation method ...

        function [E,B] = getField(obj,xk,yk,zk,tau) % return E,B fields of the RF-Gun at a certain point ...

        function plot2DField(obj,Fignumber)         % plotting 2D E-field ...

        function plot1DField(obj, Fignumber, Ez_or_Er, r_z_in_m)    % plotting either B_z or B_r at certain r or z ...
    end
end
```

Class of the solenoid-field

```matlab
classdef Sol_Field < handle
    properties
        main_filename   % filename of field distribution
        buck
        c = 299792458   % lightspeed m/s
        f = 1.3e9       % 1.3GHz
        k               % wavenumber
        B0              % maximum field strenght
        z               % z-values of field distribution
        zeta            % dim-less z-values
        Bz_norm         % normed field distribution
        Bz_1dif         % 1st diffrerentiation of B-field
        Bz_2dif         % 2nd diff
        Bz_3dif         % 3rd diff
        xk_sol_off = 0
        yk_sol_off = 0
        zk_sol_off = 0  % dim-less x,y,z-alignment of solenoid
        pitch_deg = 0
        yaw_deg = 0     % pitch,yaw of solenoid
        B_const_T       % values of a constant magnetic field
    end

    methods
        function obj = Sol_Field(mainfilename, B_0)  % constructor ...

        function obj = setB0(obj, B_0)          % setting B0 ...

        function obj = setSolPlacem(obj,x_mm,y_mm,z_mm,pitch_degrees,yaw_degrees)   % setting solenoid alignment ...

        function deriv = GetDeriv(obj,Fz,z)     % derivation method ...

        function B = getField(obj, xk,yk,zk)    % return B-field at a certain point ...

        function plot1DField(obj, Fignumber, Bz_or_Br, r_z_in_m)    % plotting either B_z or B_r at certain r or z ...

        function plot2DField(obj, Fignumber)    % plotting the 2 dimensional field ...

        function obj = set_B_const(obj, Bx_uT, By_uT, Bz_uT)    % setting constant a magnetic field ...
    end
end
```

Class of the particle tracker

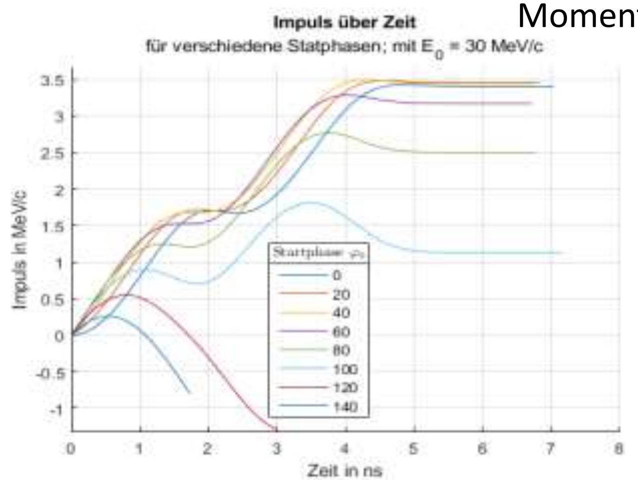
```
classdef PartTracker < handle
    properties
        c = 299792458   % lightspeed m/s
        f = 1.3e9       % 1.3GHz
        e = -1.602e-19  % elementry charge C = A*s
        m = 9.109e-31   % electron mass
        k               % wavenumber
        RFField         % RF-Field object
        SolField        % Solenoid-Field object
        PartTrack_norm  % Tracked particle in normed units
        PartTrack_unit  % Tracked particle in SI-units
        %(1,2,3):P_(x,y,z)  |   (4,5,6):x,y,z   |   (7):time
        z_end = 0.803   % in m | default end of tracking
        zeta_end        % dimesionless end of the tracking
        steps = 100     % time steps per Period
        laser_x_off_mm = 0  %
        laser_y_off_mm = 0  % x,y starting point of electron
    end

    methods
        function obj = PartTracker(RF_Field,Sol_Field)  % constructor with RF- and Sol-Field objects...

        function obj = setSteps(obj, Steps_default_100) % setting steps per period...

        function obj = setSolField(obj,Sol_Field)       % setting new solenoid field object...

        function obj = setZ_end(obj, z_End_m)           % setting end of tracking...

        function obj = setOffset(obj,x_off_mm, y_off_mm)% setting starting position...

        function obj = Tracking(obj)                    % tracking particle...

        function Y = Track_sim(obj)                     % simulation of particle (used for Tracking(obj) method)...

        function Y = GetTrack(obj)                      % returns tracked particle (PartTrack_unit)...

        function Y = GetEndState(obj)                   % returns last entries of PartTrack_unit...

        function F = GetRightside(obj,Y,tau)            % used for RK4 in Track_sim(obj)...

        function F = GetForce(obj)                      % used for plot_Overview...

        function plot_Overview(obj,Fignumber)           % plotting an Overview of the result of the simulation...

        function x = Screen_for_alignment(obj, I_vec, Input)    % returns ending postions for different solenoid currents (I_vec)+...

    end
end
```

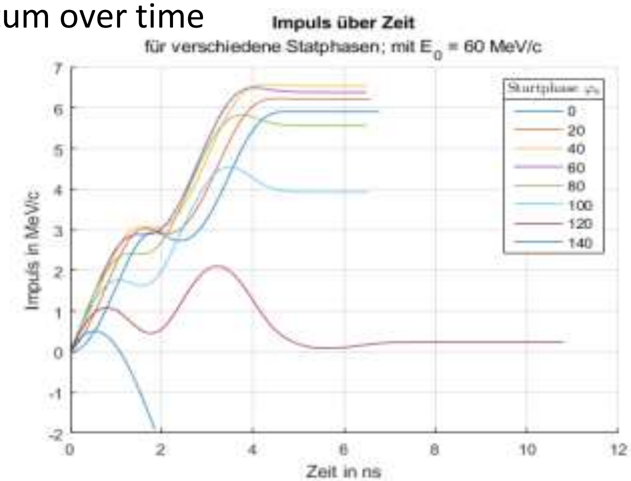
Results: 1D-dynamic

- Calculated for different φ_0 and E_0 yields:

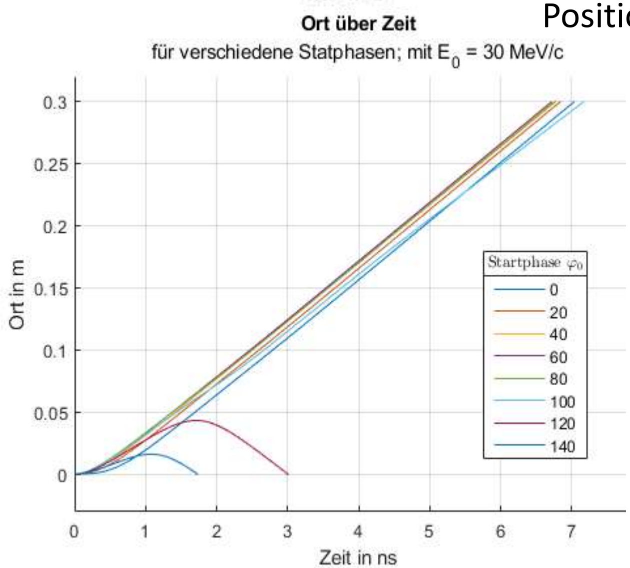
$$E_z(z,t) = E_0 \cdot E_{z,norm} \cdot \sin(\omega t + \varphi_0)$$

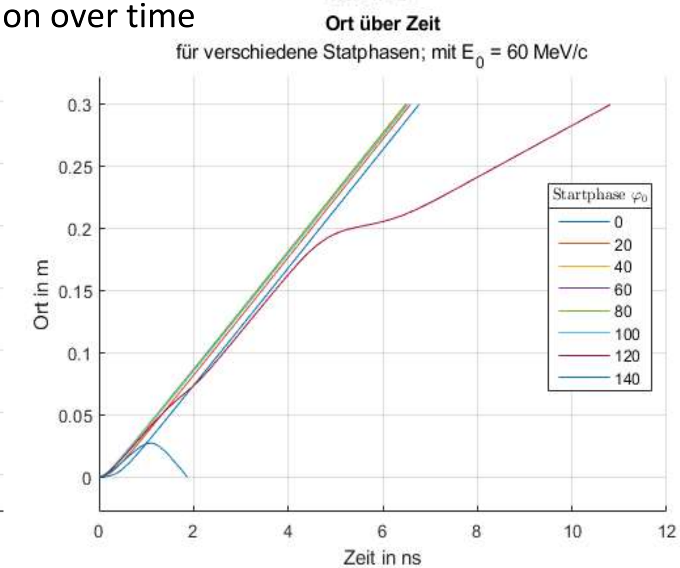
Momentum over time





Position over time

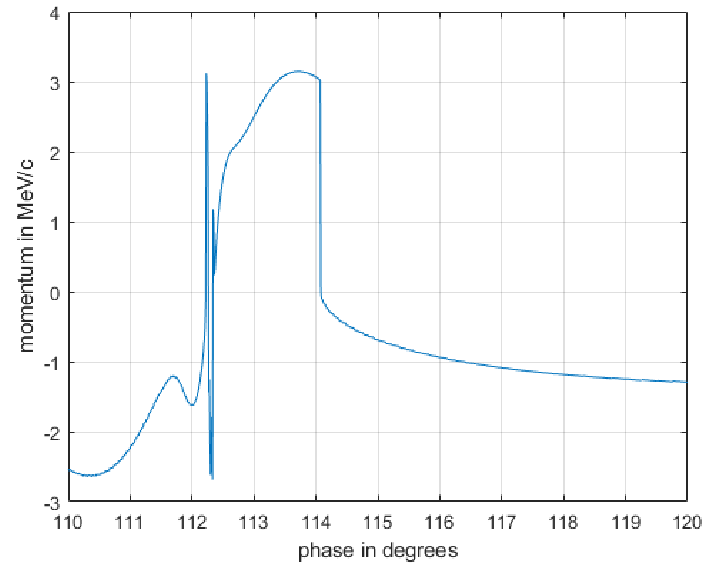


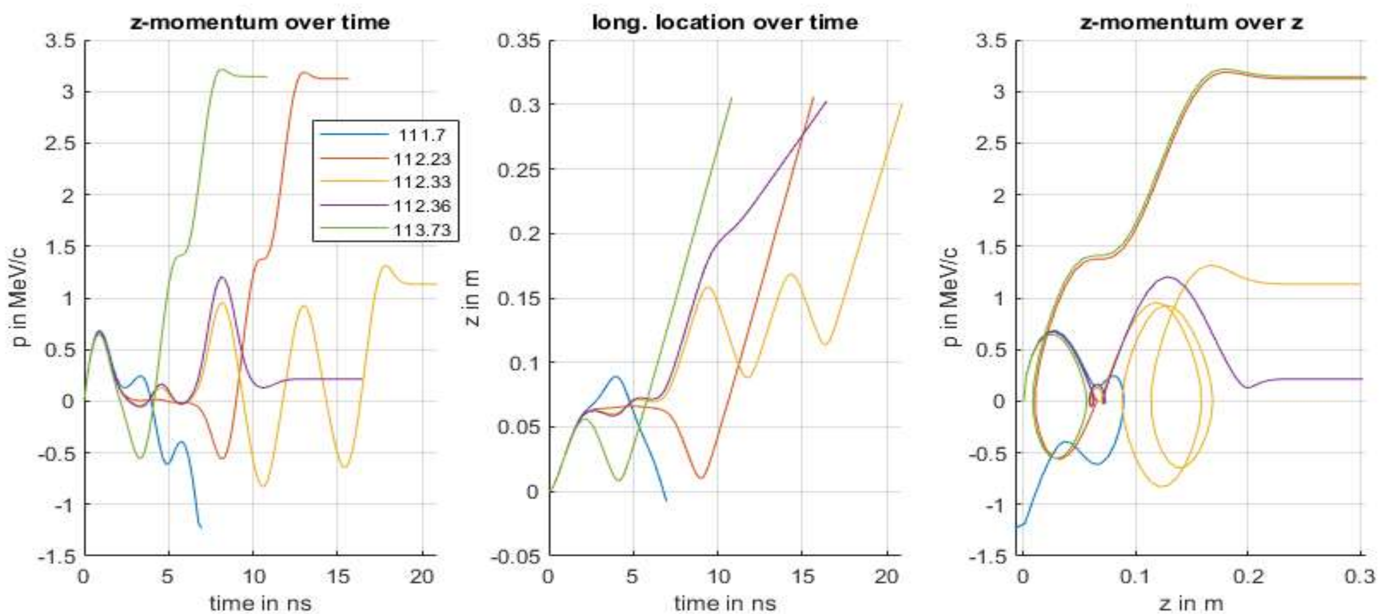


‘Exotic’ Phases

- Phases around 110 to 125 degrees (depending on gradient E_0)

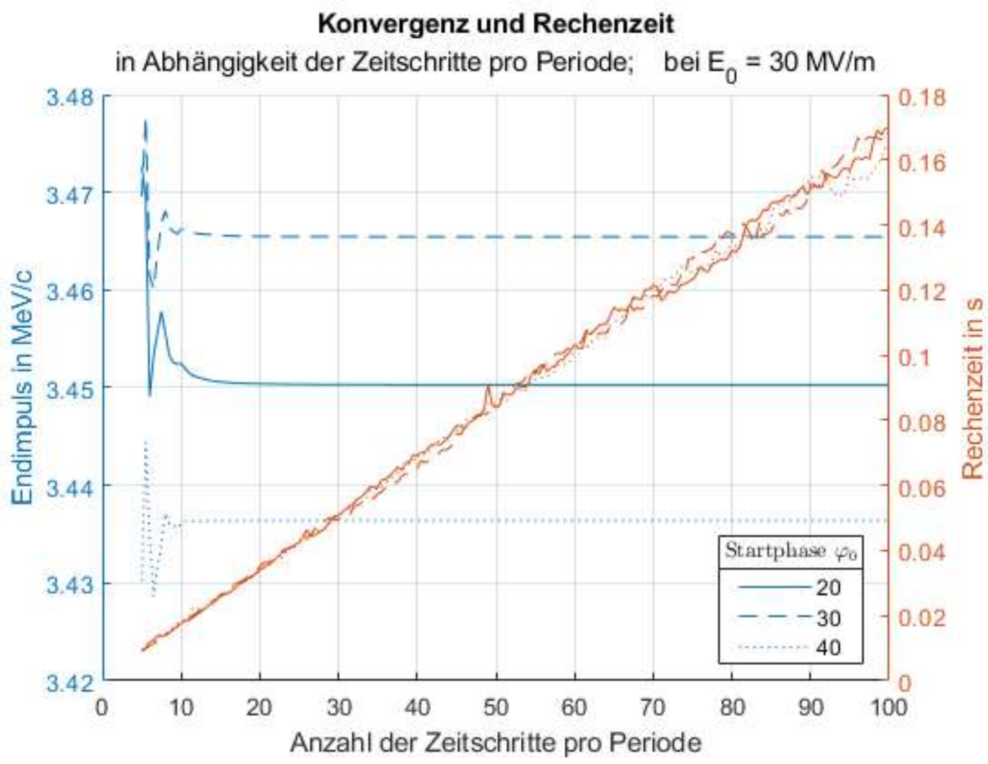
- Shown here for $E_0 = 30\,\frac{MV}{m}$:





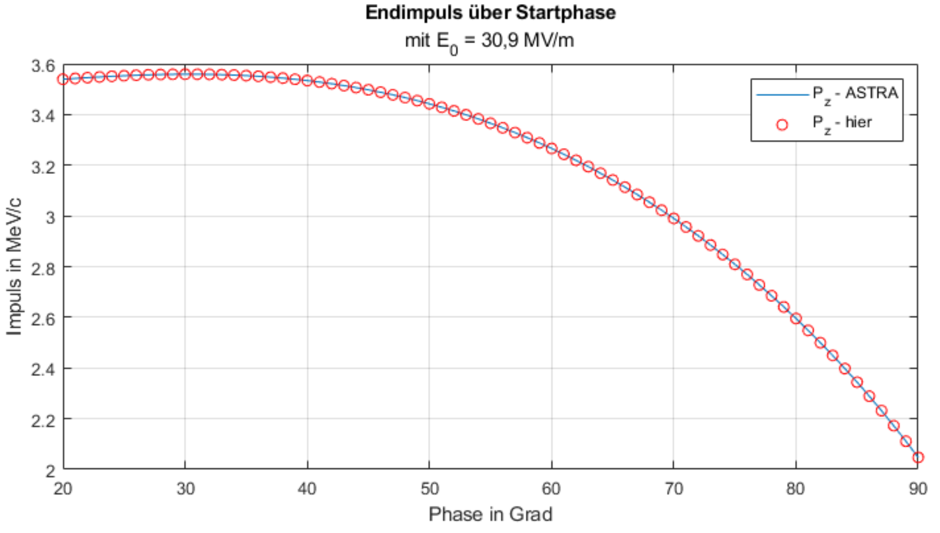
Convergence and computing-time

- Investigation of the stability and convergence of the algorithm for different φ_0:



Comparison with ASTRA: final momentum vs launch phase

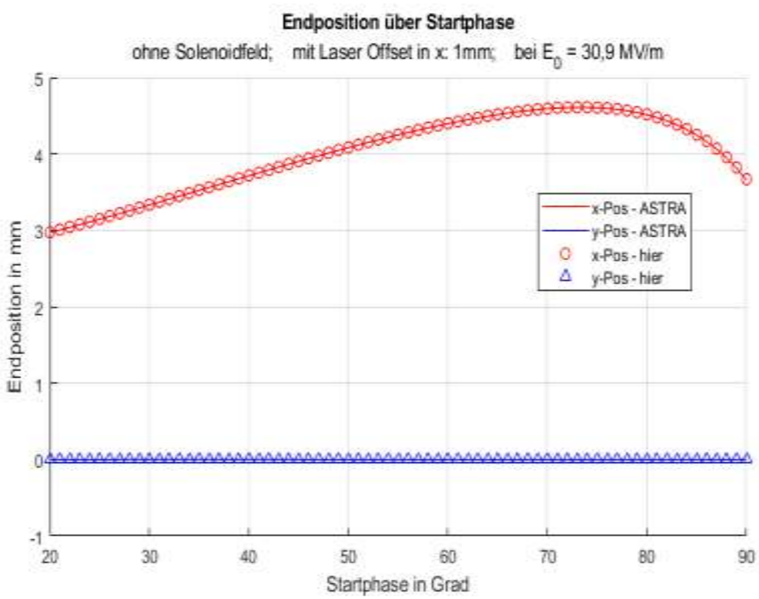
- Momentum at the end of the simulation depending on φ_0
- Ideal alignment:
  - electron was released on the RF-Gun axis
  - the solenoid axis is on and parallel towards the RF-Gun axis

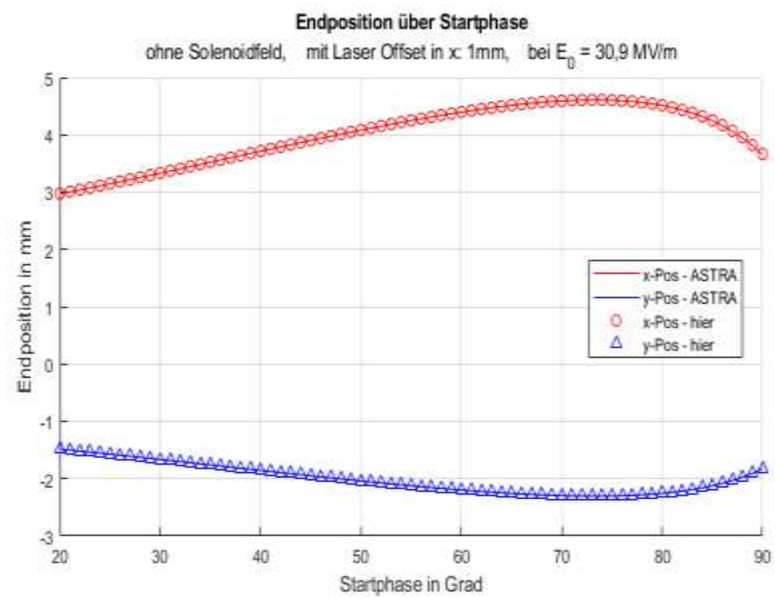


- Average difference is 0.002%

Comparison with ASTRA: final position

- Laser does not hit the cathode in the middle
- Without field of the solenoid
- Position at the end of simulation depending on φ_0 for two different starting positions

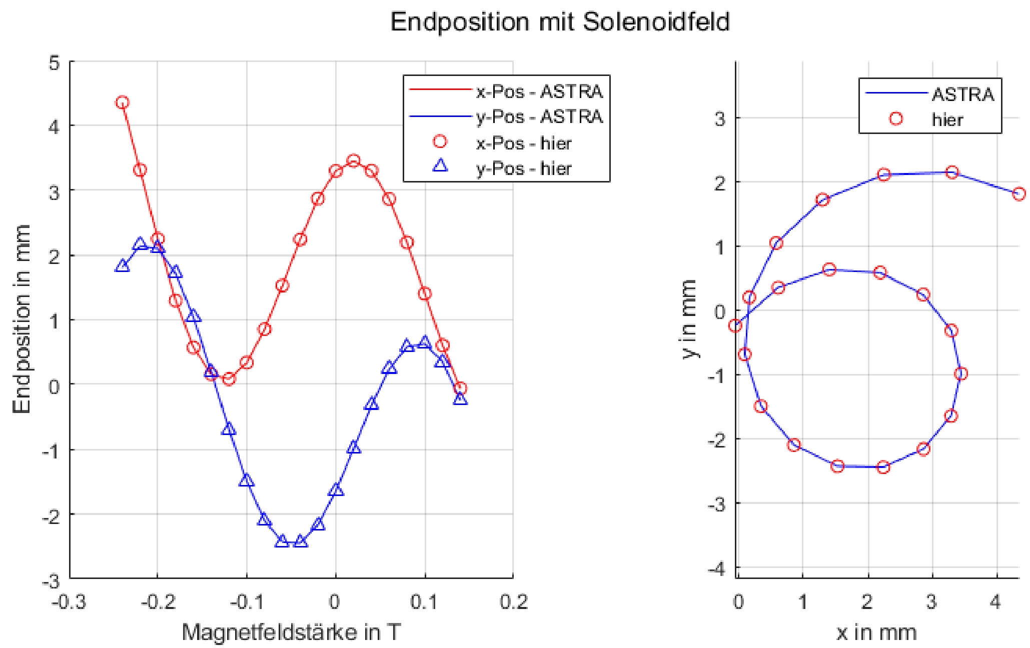




- Average difference is 0.091%

Comparison with ASTRA: final position with solenoid field

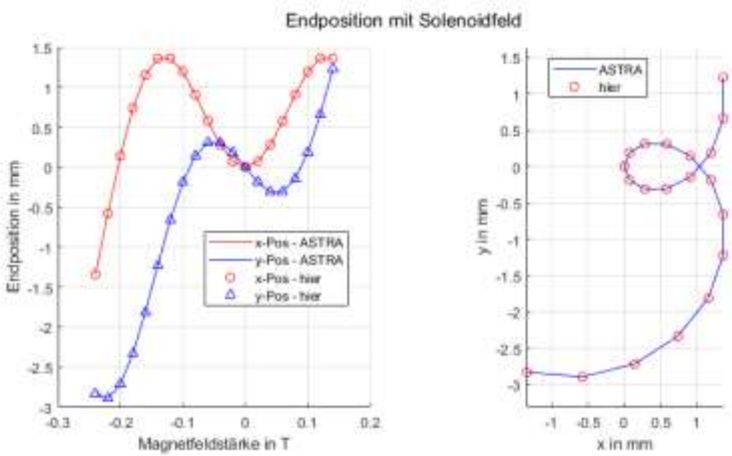
- Simulating the ending position depending on the magnetic field strength
- The solenoid is perfectly aligned
- The laser hits the cathode with an offset of $(x, y) = (1mm, -0.5mm)$



- Average difference is 0.12%

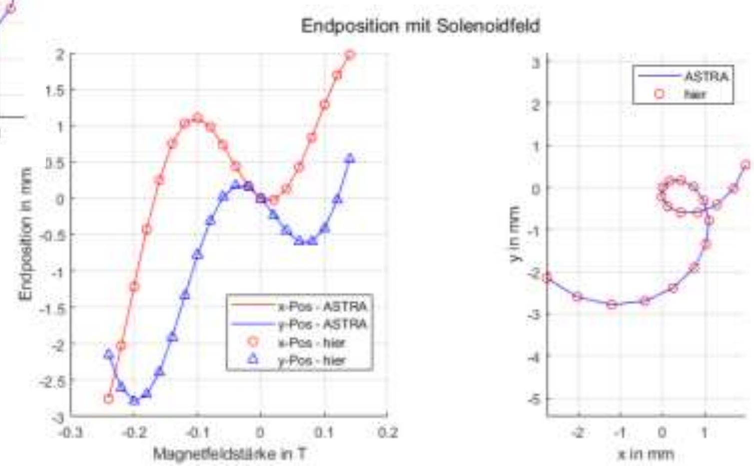
Comparison with ASTRA: final position with solenoid field

- Laser is perfectly aligned



Solenoid has an offset of:
$(x, y) = (1mm, 0mm)$

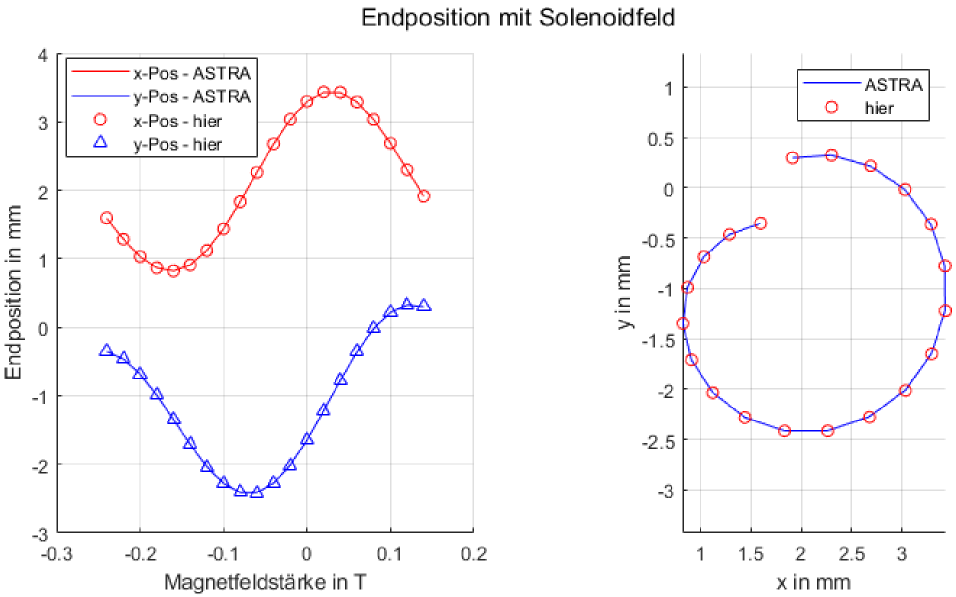
Solenoid has an offset of:
$(x, y) = (1mm, -0.5mm)$



- Average difference is 0.2%

Comparison with ASTRA: final position with solenoid field

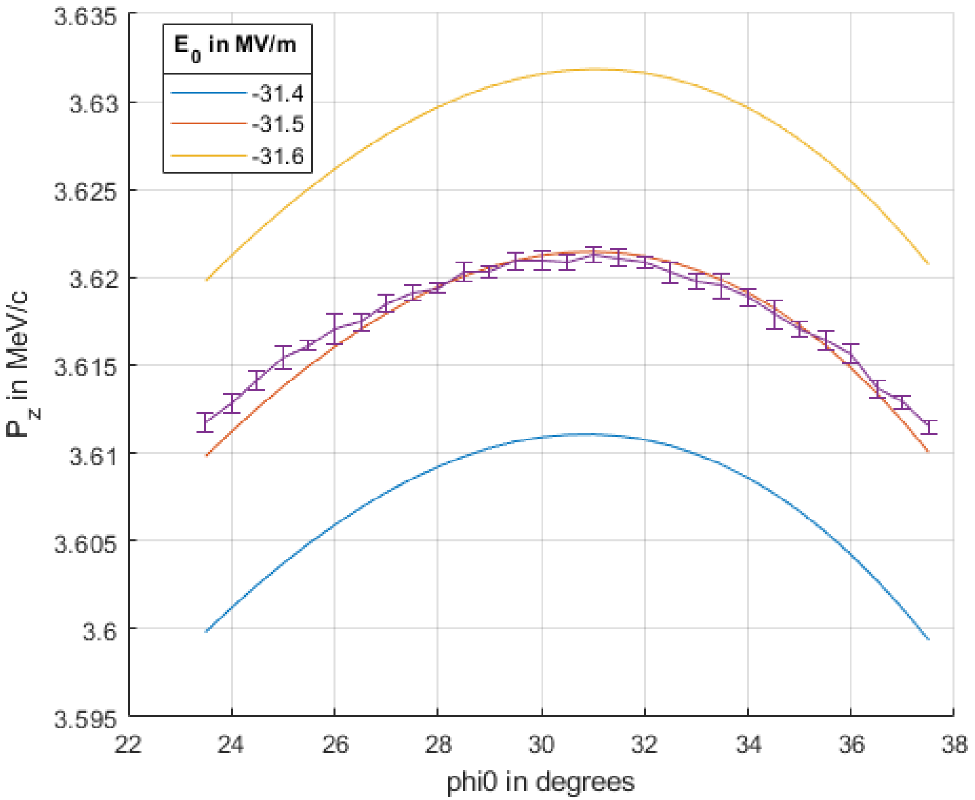
- Laser has an offset of $(x, y) = (1mm, -0.5mm)$
- Solenoid has an offset of $(x, y) = (1mm, -0.5mm)$



- Average difference is 0.3%

Fit to a measurement

- measured momentum compared to corresponding simulated momentum vs starting phase
- Measured with LEDA
- $E_0 = 31.5\,\frac{MV}{m}$ fits best



Software of the beam-based alignment

- A given measurement of final positions $\vec{x}'_M$ depending on the solenoid current I_{main} is compared to a simulated set of final positions $\vec{x}_S$ with a certain alignment vector $\vec{A}$
  (laser spot-offset, solenoid-offset, solenoid-pitch and yaw, constant magnetic field)
- The magnetic field strength is given by:
  $$B_0 = I_{main} \cdot 5.871 \cdot 10^{-4} \frac{T}{A}$$
- The center of the monitor for the measurementis not aligned to the axis, hence a monitor offset has to be calculated by:
  $$\vec{x}_{off} = \frac{1}{\sum w_i} \sum_{i=1}^{N} \vec{w}_i \left(\vec{x}_{M,i} - \vec{x}_{S,i}\right),$$
- with $\vec{w}_i$ being a weight of a certain measured point i
- That offset will be used to adjust the monitor on the axis by:
  $$\vec{x}_M = \vec{x}'_M - \vec{x}_{off}$$

Software of the beam-based alignment

- To find the alignment vector $\vec{A}$ a goal function $F\left(\vec{A}\right)$ is minimized:

$$F\left(\vec{A}\right) = \sum_{i=1}^{N} \{w_{x,i}\left(x_{M,i} - x_{S,i}(\vec{A})\right)^2 + w_{y,i}\left(y_{M,i} - y_{S,i}(\vec{A})\right)^2\}$$

- The weight $\vec{w}_i$ of a certain measured point i reads:

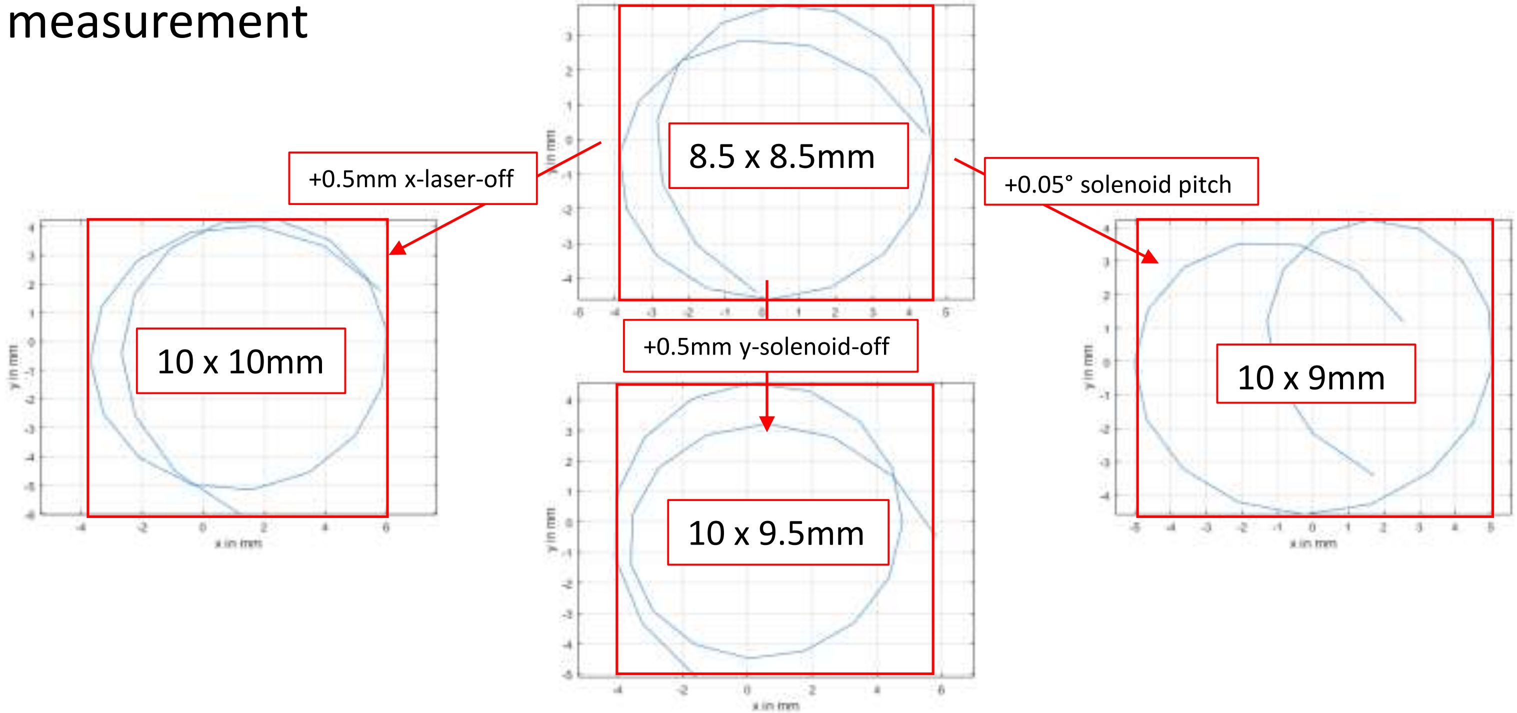
$$w_{x,i} = \exp\left(-\frac{\epsilon_{x,i}^2}{2\sigma_x^2}\right) \quad \text{and} \quad w_{y,i} = \exp\left(-\frac{\epsilon_{y,i}^2}{2\sigma_y^2}\right)$$

- With ϵ_i being the statistical error of a single measurement and σ being the root mean square of all measured errors

- The minimization of the goal function is done by the MATLAB function *fminsearch* which minimizes a multivariable function via the Downhill-Simplex-method and gives back the coordinates of the calculated minimum

Normalization

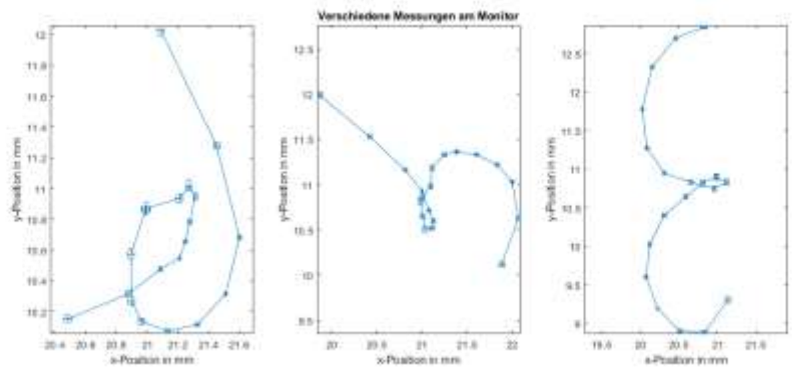
- To equal the impact of each alignment variable on the goal function value, each variable is normalized
- That normalization was determined by the impact on the monitor measurement



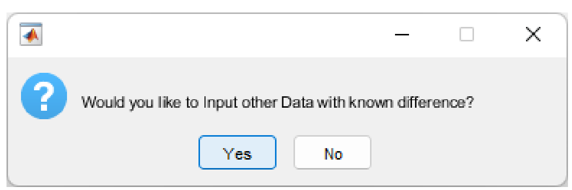
- 0.5 mm laser or solenoid offset lead to same beam misalignment as 0.05° solenoid angle

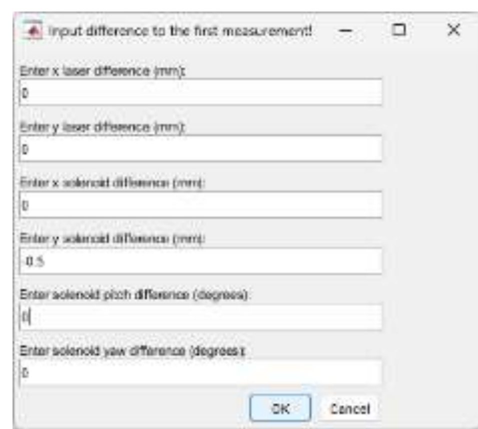
Structure of the software: Input

- Software starts by choosing the file with the measured data points which has to include the solenoid current, and the positions at the screen with statistical errors
- Those measurements may look as follows:



- For better accuracy there is an option to chose multiple measurements with a **known difference** in the alignment





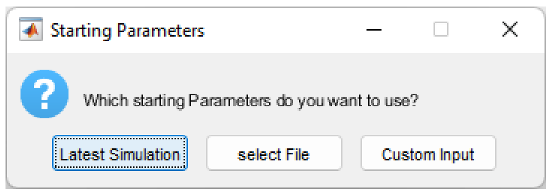
Structure of the software: Input

- Then the gradient and the phase of the measurement has to be set

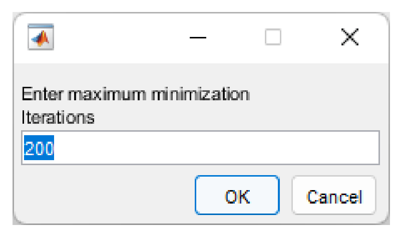
- The free choice of optimisation variables

Structure of the software: Input

- The alignment at the start of the minimization can be read from a file, e.g. the last measurement performed, or inserted manually

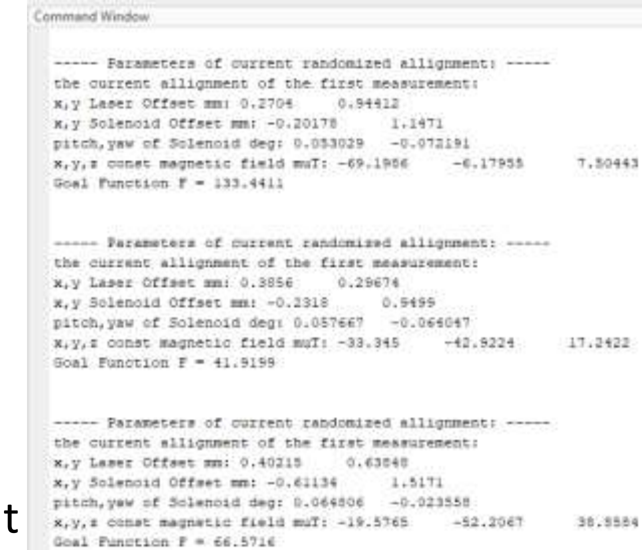


- Finally the maximum number of minimization iterations for *fminsearch* has to be set



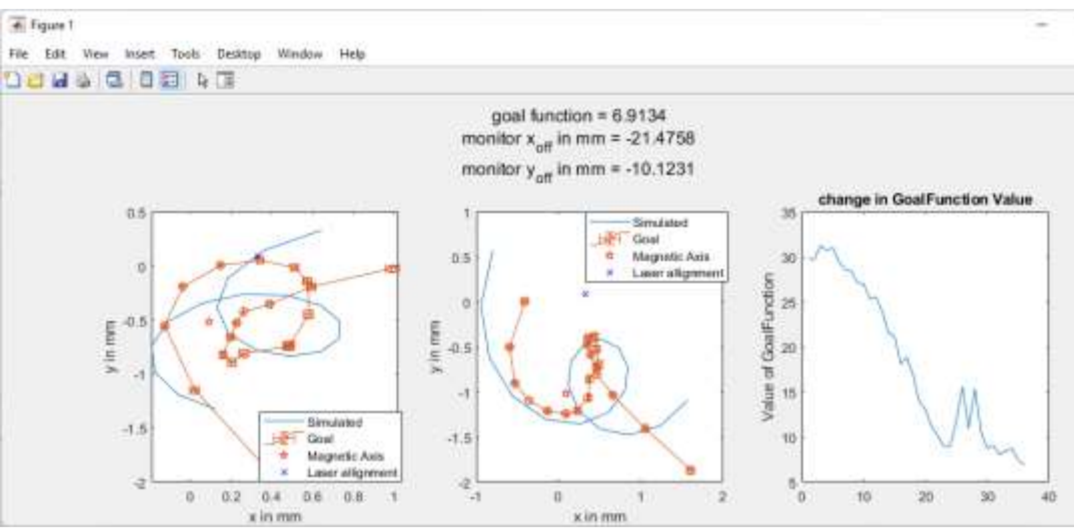
Structure of the software: optimization algorithm

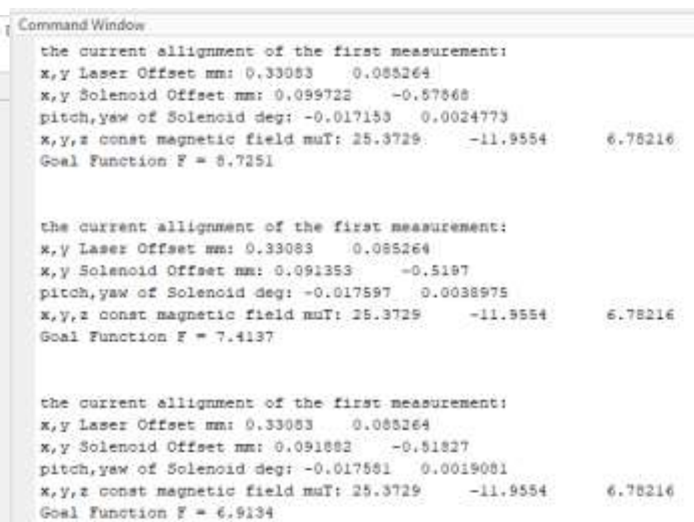
- Afterwards 10, based on starting positions, randomized alignments are simulated, results are shown on the console



```
----- Parameters of current randomized allignment: -----
the current allignment of the first measurement:
x,y Laser Offset mm: 0.2704      0.94412
x,y Solenoid Offset mm: -0.20178      1.1471
pitch,yaw of Solenoid deg: 0.053029   -0.072191
x,y,z const magnetic field muT: -69.1986      -6.17955      7.50443
Goal Function F = 133.4411


----- Parameters of current randomized allignment: -----
the current allignment of the first measurement:
x,y Laser Offset mm: 0.3856      0.29674
x,y Solenoid Offset mm: -0.2318      0.9499
pitch,yaw of Solenoid deg: 0.057667   -0.064047
x,y,z const magnetic field muT: -33.345      -42.9224      17.2422
Goal Function F = 41.9199


----- Parameters of current randomized allignment: -----
the current allignment of the first measurement:
x,y Laser Offset mm: 0.40215      0.63848
x,y Solenoid Offset mm: -0.61134      1.5171
pitch,yaw of Solenoid deg: 0.064806   -0.023558
x,y,z const magnetic field muT: -19.5765      -52.2067      38.8884
Goal Function F = 66.5716
```

- The best of those is being used as a start for the minimization process (to build an initial simplex)
- During the minimization the last simulated alignment is being shown

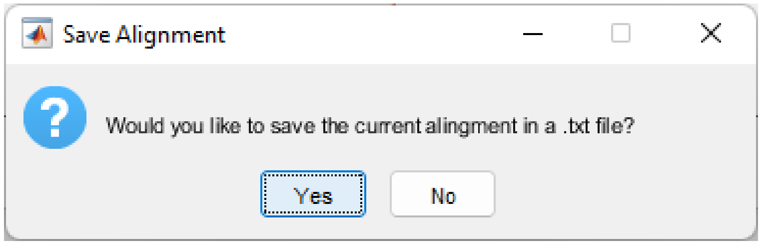


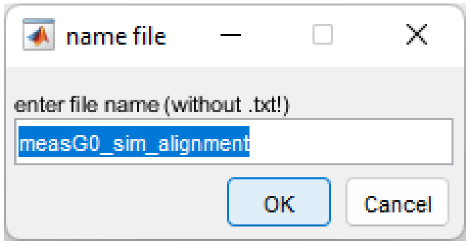


```
the current allignment of the first measurement:
x,y Laser Offset mm: 0.33083    0.085264
x,y Solenoid Offset mm: 0.099722    -0.57868
pitch,yaw of Solenoid deg: -0.017153   0.0024773
x,y,z const magnetic field muT: 25.3729      -11.9554      6.78216
Goal Function F = 8.7251


the current allignment of the first measurement:
x,y Laser Offset mm: 0.33083    0.085264
x,y Solenoid Offset mm: 0.091353     -0.5197
pitch,yaw of Solenoid deg: -0.017597   0.0038975
x,y,z const magnetic field muT: 25.3729      -11.9554      6.78216
Goal Function F = 7.4137


the current allignment of the first measurement:
x,y Laser Offset mm: 0.33083    0.085264
x,y Solenoid Offset mm: 0.091882    -0.51827
pitch,yaw of Solenoid deg: -0.017581   0.0019081
x,y,z const magnetic field muT: 25.3729      -11.9554      6.78216
Goal Function F = 6.9134
```

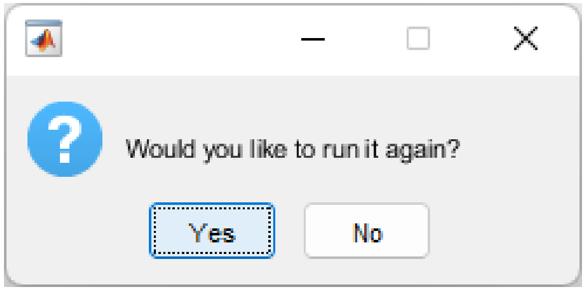
Structure of the software: results

- After a minimization cycle there is the option to save the simulated alignment in a .txt file



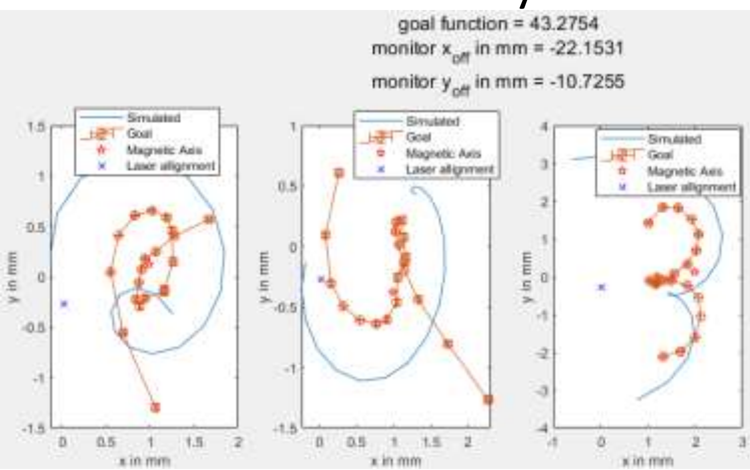


- Finally there is the option to start another cycle of minimization using a different selection of alignment variables to be minimised



Example for Application

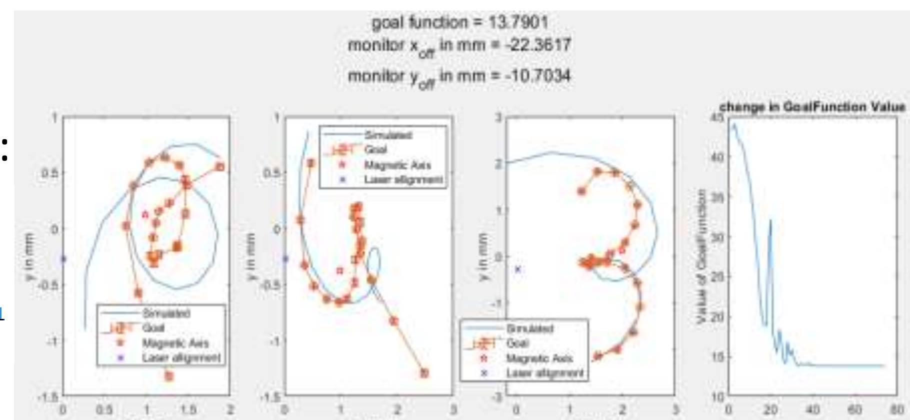
- Three measurements
  - In the second one the solenoid is moved by $-0.5mm$ in y-direction compared to the first
  - In the third one the solenoid is moved by $+1mm$ in x-direction compared to the first
- After the randomly simulated alignments the best one is:




```
----- Parameters of current randomized allignment: -----
the current allignment of the first measurement:
x,y Laser Offset mm: 0.021136    -0.26841
x,y Solenoid Offset mm: 0.9889     0.12406
pitch,yaw of Solenoid deg: 0.014331  -0.0083588
x,y,z const magnetic field muT: -13.2563      48.7982     -46.2261
Goal Function F = 43.2754
```

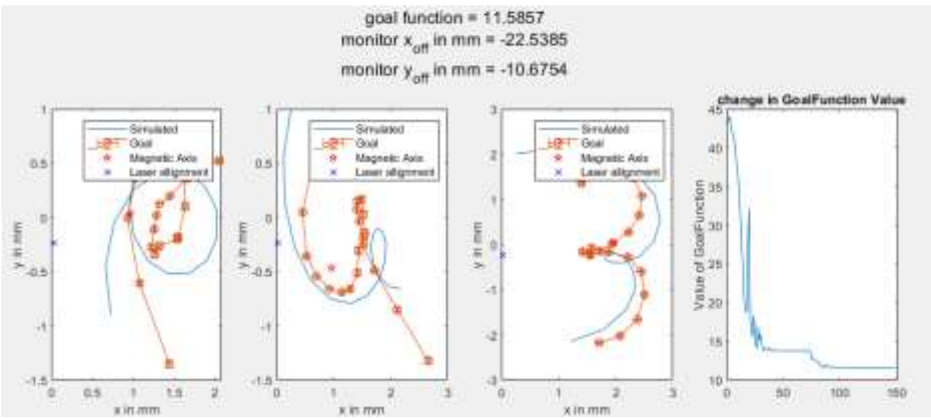
- After minimization of the solenoid angles:

```
the current allignment of the first measurement:
x,y Laser Offset mm: 0.021136    -0.26841
x,y Solenoid Offset mm: 0.9889     0.12406
pitch,yaw of Solenoid deg: 0.011218    0.021441
x,y,z const magnetic field muT: -13.2563      48.7982     -46.2261
Goal Function F = 13.7901
```



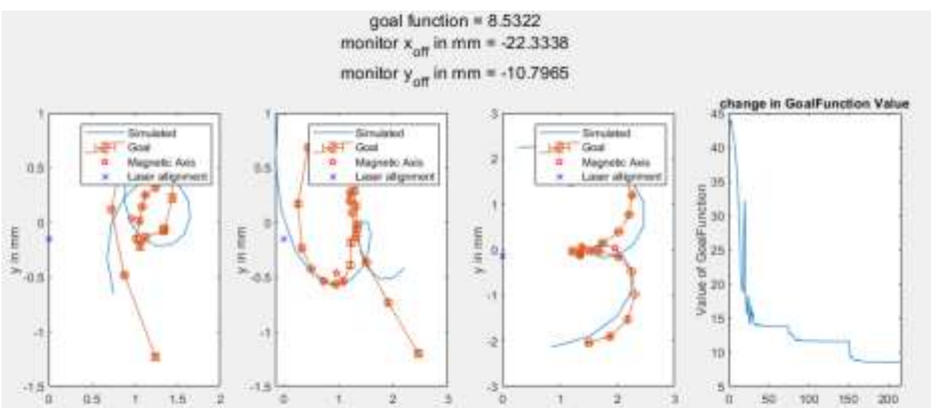
Application

- After minimization of the solenoid offset:




```
the current allignment of the first measurement:
x,y Laser Offset mm: 0.025228    -0.23268
x,y Solenoid Offset mm: 0.96697    0.035533
pitch,yaw of Solenoid deg: 0.0098634    0.021353
x,y,z const magnetic field muT: -12.4052      57.6137     -53.9982
Goal Function F = 11.5857
```

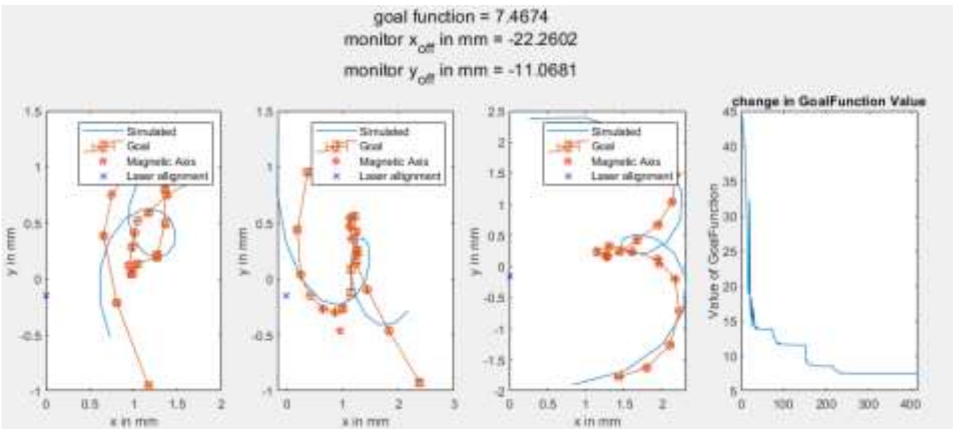
- After minimization of the laser offset:




```
the current allignment of the first measurement:
x,y Laser Offset mm: -0.0048696    -0.15211
x,y Solenoid Offset mm: 0.96409    0.035199
pitch,yaw of Solenoid deg: 0.0084933    0.019996
x,y,z const magnetic field muT: -12.9381      50.5098      -59.148
Goal Function F = 8.5322
```

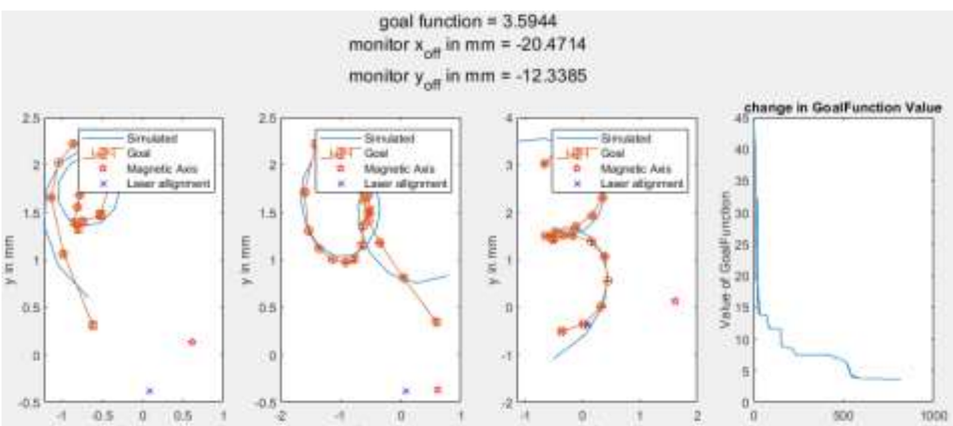
Application

- After simulating an additional constant magnetic field:




```
the current allignment of the first measurement:
x,y Laser Offset mm: -0.0049604    -0.14759
x,y Solenoid Offset mm: 0.96202    0.037277
pitch,yaw of Solenoid deg: 0.0091015    0.021082
x,y,z const magnetic field muT: -25.2439      46.7069     0.355824
Goal Function F = 7.4674
```

- Finally all variables are minimized simulaneously:




```
the current allignment of the first measurement:
x,y Laser Offset mm: 0.092856    -0.38894
x,y Solenoid Offset mm: 0.61486     0.13204
pitch,yaw of Solenoid deg: 0.039951   -0.023553
x,y,z const magnetic field muT: -89.9886     -20.7286      1.47509
Goal Function F = 3.5944
```

Summary

- A code for the beam dynamics of a reference particle in a RF-Photogun has been developed
- The cross-check with ASTRA shows very good agreement
- Potential sources of misalignment have been implemented
  - Laser spot-offset, Solenoid-offset, Solenoid pitch and yaw, constant magnetic field
- A goal function based on weighted measurements has been introduced
- The program has the option for subsequent simulations to exploit measurements with known differences in alignment (e.g. solenoid movements)
- The code has been applied to experimental data sets for two solenoid test movements (basic + 2 test movements), resulting in a good fit
- The resulting misalignment:
  - Laser-off. in mm: (0.09,-0.39); Solenoid-off. in mm: (0.61,0.13); Solenoid pitch and yaw in deg.: (0.04,-0.02)
  - Constant magnetic field in µT: (-90,-20,1.4)
- The package is prepared for practical use

Thank you for your attention!

3-dimensional fields of the RF-Gun

- Applying polynomial expansion in r to Maxwell's equations in cylindrical coordinates (r, θ, z), the components of $\vec{E}$ and $\vec{B}$ can be derived

$$E_z(t,r,z) = \left[E_z(r=0,z) - \frac{r^2}{4}\left(E_z''(r=0,z) + \frac{\omega^2}{c^2}E_z(r=0,z)\right) + O(r^4)\right]\sin(\omega t + \varphi_0)$$

$$E_r(t,r,z) = \left[-\frac{r}{2}E_z'(r=0,z) + \frac{r^3}{16}\left(E_z'''(r=0,z) + \frac{\omega^2}{c^2}E_z'(r=0,z)\right) - O(r^4)\right]\sin(\omega t + \varphi_0)$$

$$B_\theta(t,r,z) = \frac{\omega}{c^2}\left[\frac{r}{2}E_z(r=0,z) - \frac{r^3}{16}\left(E_z''(r=0,z) + \frac{\omega^2}{c^2}E_z(r=0,z)\right) + O(r^4)\right]\cos(\omega t + \varphi_0)$$

- With the cylindrical coordinates and their unit vectors expressed in cartesian coordinates:

$$r = \sqrt{x^2 + y^2}$$
$$\theta = \arctan\left(\frac{x}{y}\right)$$
$$z = z$$

$$\hat{e}_r = \hat{e}_x \cos(\theta) + \hat{e}_y \sin(\theta)$$
$$\hat{e}_\theta = -\hat{e}_x \sin(\theta) + \hat{e}_y \cos(\theta)$$
$$\hat{e}_z = \hat{e}_z$$

- The $\vec{E}$ and $\vec{B}$ fields can be expressed as:

3-dimensional fields of the RF-Gun

$$\vec{E} = E_0 \begin{pmatrix} -\frac{x}{2}\left[E'_{z,norm}(z) + \frac{x^2+y^2}{8}\left(E''_{z,norm}(z) + \frac{\omega^2}{c^2}E_{z,norm}(z)\right)\right] \\ -\frac{y}{2}\left[E'_{z,norm}(z) + \frac{x^2+y^2}{8}\left(E''_{z,norm}(z) + \frac{\omega^2}{c^2}E_{z,norm}(z)\right)\right] \\ E_{z,norm}(z) - \frac{x^2+y^2}{4}\left(E''_{z,norm}(z) + \frac{\omega^2}{c^2}E_{z,norm}(z)\right) \end{pmatrix} \cdot \sin(\tau + \varphi_0)$$

And

$$\vec{B} = E_0 \frac{\omega}{c^2} \begin{pmatrix} -\frac{y}{2}\left[E_{z,norm}(z) - \frac{x^2+y^2}{8}\left(E''_{z,norm}(z) + \frac{\omega^2}{c^2}E_{z,norm}(z)\right)\right] \\ -\frac{y}{2}\left[E_{z,norm}(z) - \frac{x^2+y^2}{8}\left(E''_{z,norm}(z) + \frac{\omega^2}{c^2}E_{z,norm}(z)\right)\right] \\ 0 \end{pmatrix} \cdot \cos(\tau + \varphi_0)$$

3-dimensional field of the Solenoid

- Similar to the field of the RF-Gun the Field of the Solenoid can be expressed with:

$$B_z(r,z) = B_z(r=0,z) - \frac{r^2}{4} B_z''(r=0,z) + O(r^4)$$

$$B_r(r,z) = -\frac{r}{2} B_z'(r=0,z) + \frac{r^3}{16} B_z'''(r=0,z) + O(r^5)$$

- Which leads to:

$$\vec{B} = B_0 \begin{pmatrix} -\frac{x}{2}\left[B_{z,norm}'(z) + \frac{x^2+y^2}{8} B_{z,norm}'''(z)\right] \\ -\frac{y}{2}\left[B_{z,norm}'(z) + \frac{x^2+y^2}{8} B_{z,norm}'''(z)\right] \\ B_{z,norm}(z) - \frac{x^2+y^2}{4} B_{z,norm}''(z) \end{pmatrix}$$