Investigations on invasiveness of electron probe diagnostic for PWFA

Florian König, University of Hamburg, Germany
Supervisor: Pardis Niknejadi

September 4, 2019

Abstract

Plasma Wakefield Acceleration (PWFA) has become a promising candidate for affordable and compact future particle accelerators, but it comes with high demands in terms of beam quality and stability. The FLASHForward experiment at DESY aims to produce such high-quality electron beams at GeV-scale within a plasma cell of a few centimeters. In the frame of this work, the method of Femtosecond innovative RelativistiC Electron (FiRCE) probing as a beam diagnostic was investigated concerning its invasiveness on the driving beam. Therefore, the beam-plasma interaction was simulated in OSIRIS and a Matlab script for analysis of the impact on the driving beam in terms of ellipse parameters was written.

Contents

1. Introduction and Motivation

The use of high energy physics and particle accelerators enables the scientific understanding of nature on time and length scales beyond the human perception. Besides their ability to test the subatomic structures of our world in collider experiments, the acceleration of particles in synchrotron rings or free-electron lasers (FELs) can be used as radiation sources that find application in modern medicine, industry and research. However, the aim of continuously increasing time and length resolution implies a need for increasing energies.

Whereas conventional acceleration based on RF Cavities is ultimately limited to electric field gradients of 0.1GeV/m, due to electric or thermal breakdown, the enormous electric fields produced by separation of electrons and ions in a dense plasma, exceed these gradients by a factor of up to $100 - 1000$. Hence, plasma-based particle accelerators would allow for a severe reduction of the acceleration section, f.e. in case of the European XFEL from several hundred meters down to several meters.

The Future-ORiented Wakefield Accelerator Research and Development (FLASHForward) is a project at the DESY which aims to produce such high-quality electron beams at GeV-scale within a plasma cell of a few centimeters.
With all the advantages of producing ultra-relativistic, quasi-monoenergetic and ultrashort electron bunches, plasma-based acceleration comes with the price of very high demands in terms of beam stability and quality.
Therefore, the development of tools for beam diagnostics that are capable of characterizing ultra-short bunches as well as their interaction with the plasma is an important part of research. Whilst most beam diagnostics only analyse the beam several meters apart from their production, the Femtosecond innovative RelativistiC Electron (FiRCE) probe can be used right at the electron-plasma interaction point, thus directly analyse the wakefield acceleration process.

As seen in Figure 2 a probe beam of relatively high divergence is produced orthogonally to the main beam and scans through the driving electron bunch and its wakefield.
After the interaction, the probe beam is analysed and the properties of main bunch and plasma can be reconstructed. Even though the probe beam is usually not denser than 1% of the main beam's density (of $\sim 10^{19}\mathrm{cm}^{-3}$), the question arises whether the crossing

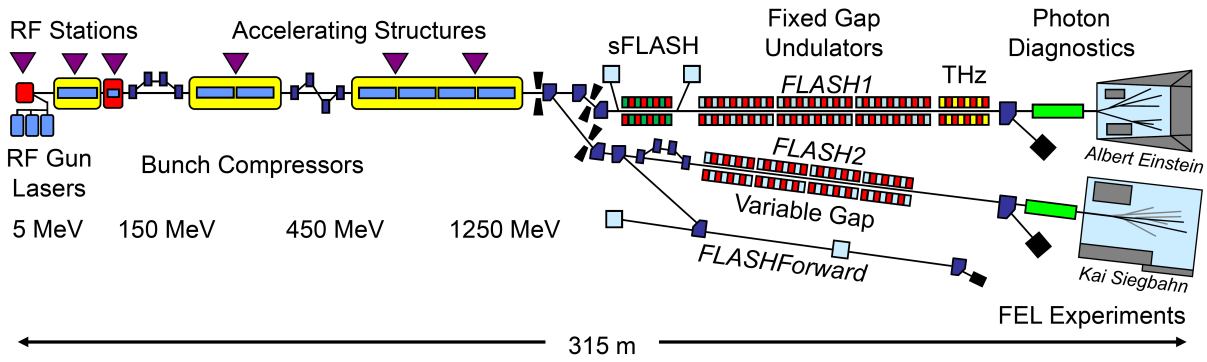


Figure 1: Schematic layout of the FLASH facility housing the two main SASE beamlines and the FLASHForward plasma acceleration experiment. [1]

does affect the quality of the main beam or not. To investigate the invasiveness of this method, various simulations were run in root-based PIC code OSIRIS and a Matlab script for analysis of the impact on the beam ellipse parameters was written.

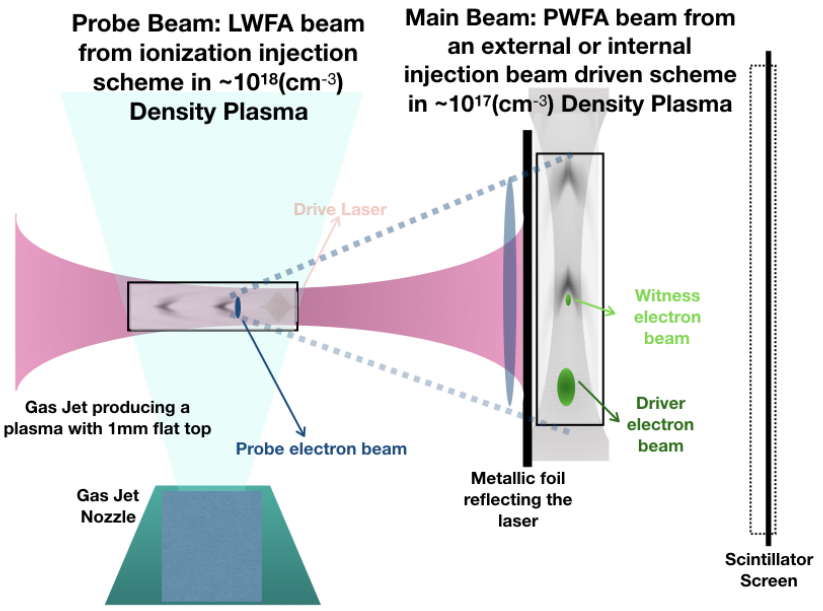


Figure 2: Simplified model of the FiRCE experimental setup. [2]

2. Theoretical Background

2.1. Transverse beam dynamics

Speaking of an ensemble of particles, or so to say, a particle bunch or beam, there is a very useful approach of describing the particles as a whole ensemble in 6D phase space. Each particle is characterised by its position in the three dimensional space as well as the corresponding momenta in these dimensions.
One defines the transverse deposition and divergence of a particle in respect to the nominal path of an ideal particle (also reference orbit) as x and $x' = p_x/p_0$. The spread of the distribution, called emittance ε_x, follows as

$$\varepsilon_x = \sigma_x \cdot \sigma'_x = \sqrt{\overline{x^2} \cdot \overline{x'^2} - \overline{xx'^2}}. \tag{1}$$

The emittance forms an ellipse in phase-space and most importantly it is invariant under action of conservative forces. The ellipse is characterised by the TWISS-Parameters $\alpha(s), \beta(s), \gamma(s)$:

$$\sigma_x(s) = \sqrt{\overline{x^2}} \equiv \sqrt{\epsilon\beta(s)}$$

$$\sigma'_x(s) = \sqrt{\overline{x'^2}} \equiv \sqrt{\epsilon\gamma(s)}$$

$$\overline{xx'} \equiv -\alpha(s)\varepsilon$$

$$\varepsilon = \gamma x^2 + 2\alpha x x' + \beta x'^2 \quad \text{emittance}$$

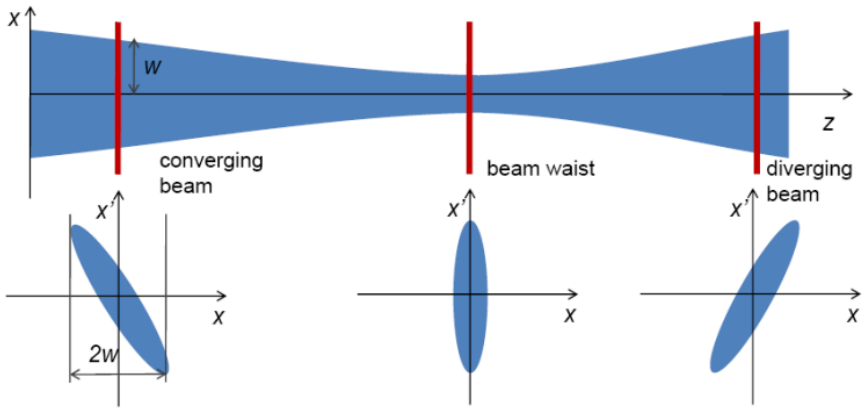


Figure 3: Transformation of phase space ellipse along the beamline. Note that the area $\pi\varepsilon_x$ remains constant. [3]

2.2. PIC code

Particle-in-cell (PIC) codes are a very powerful numerical method for modelling complex kinetic and relativistic plasma phenomena. The ability of being highly parallelisable enables a very resource efficient way for fully 3D investigations on beam-plasma interactions at very high resolutions of time and length scale. The physical foundation of such a PIC code is a Maxwell-Vlasov system as usually used for collisionless plasma physics. Since a full and direct approach would be way too impractical for particle densities as given in plasma acceleration, the particle-in-cell method introduces so called macro-particles that form larger chunks of charge. Thereby, the particle distribution function as well as the particle trajectories are numerically discretised. The Eulerian information of the system, namely the electric field $\mathbf{E}(\mathbf{r}, t)$, the magnetic field $\mathbf{B}(\mathbf{r}, t)$, and the charge- and current-density $\rho(\mathbf{r}, t)$ and $\mathbf{J}(\mathbf{r}, t)$, is evolved on a spatial grid. [4]
This allows for a very efficient investigation of complex kinetic and relativistic plasma phenomena.

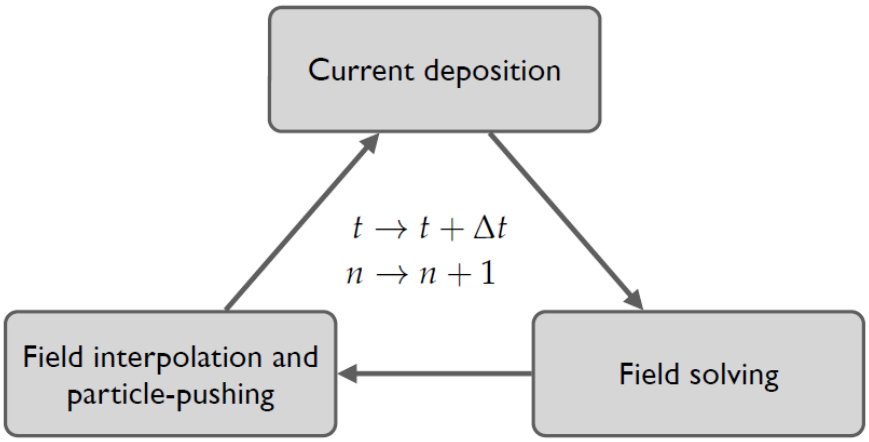


Figure 4: Main numerical time-integration cycle in PIC codes. [4]

3. Simulation in OSIRIS

OSIRIS is a 3D, fully explicit electrodynamic and relativistic PIC code, which is massively parallelized and has shown scalability to over 106 cores on supercomputers of the highest performance class. [5]

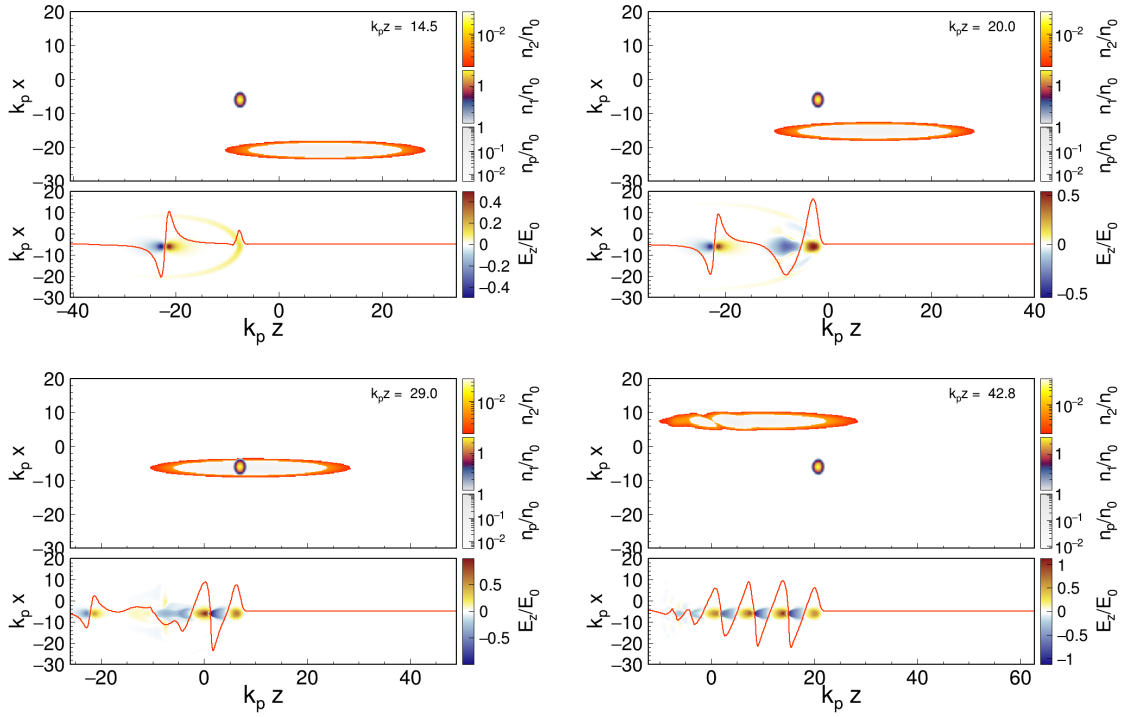


Figure 5: Snapshots at timesteps t = {21, 29, 42, 62} of example simulation ($n_e = 3 \cdot n_p$) created with OSIRIS.

4. Method of Investigation

For the following simulations, the aim was to design the driving beam in accordance with the realistic beam parameters of the FLASHForward experiment. Therefore, the following parameters were set and hold constant throughout all of the simulations.

- $n_p = 3.0 \cdot 10^{18}$ cm^{-3} - Plasma Density
- $E_{bunch} = 500$ MeV - Bunch Energy
- $\Delta E/E = 0.1\%$ - Energy Spread
- $n_{probe} = 0.05 \cdot n_p$ - Electron Probe Density

Within the scope of this work, the probe beam and plasma density remained unchanged, whereas the focus lied on varying the properties of the driving bunch. Hence, the main beam density was altered between [3.0, 4.5, 7.5, 9.0, 12.0] x n_p, corresponding to roughly [200pC, 300pC, 500pC, 600pC, 800pC] and therefore realistic values for the driving bunch at FLASHForward as well as in the regime of [0.3, 0.5, 1.0, 2.0, 3.0] x n_p, which

corresponds to charges that a typical witness bunch would carry [20pC, 33pC, 67pC, 133pC, 200pC].
For each density, the simulation was run once with a crossing probe beam as described above and then rerun with the same parameters but without any probe beam. Then, the beam ellipse parameters were analysed under the aspect of change due to interaction with the probe beam to investigate the question whether this method appears to be invasive or not.

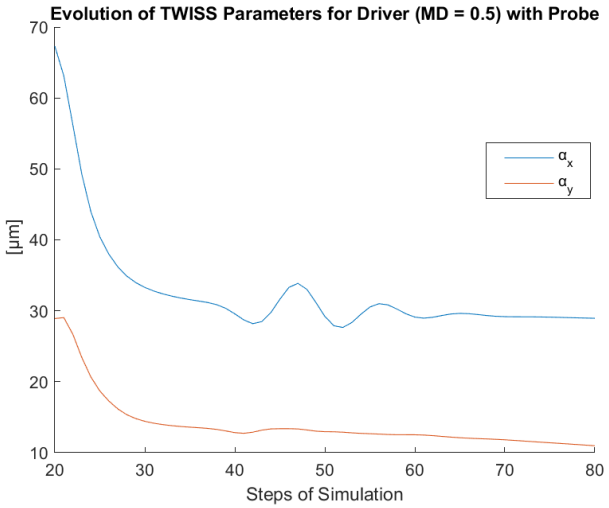
5. Results

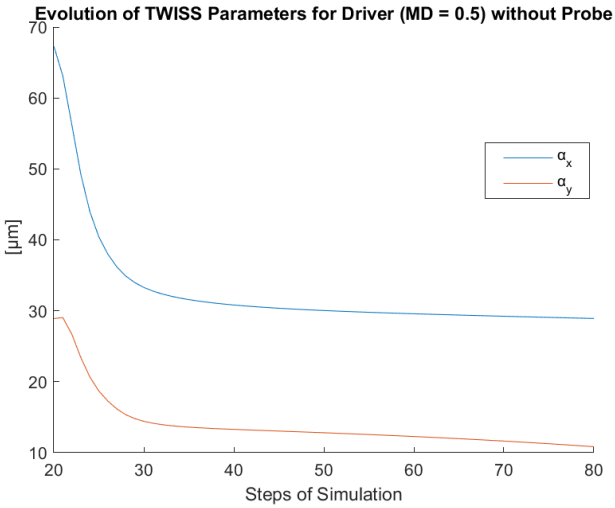
5.1. Evolution of TWISS parameters

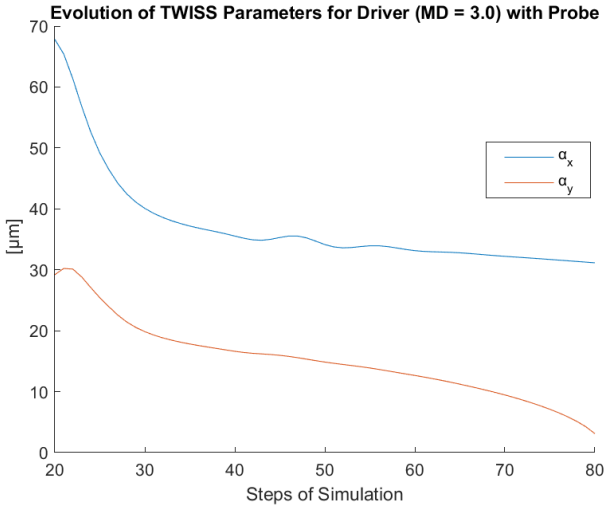
during the simulation and with probing
impact of scanning through the driver bunch

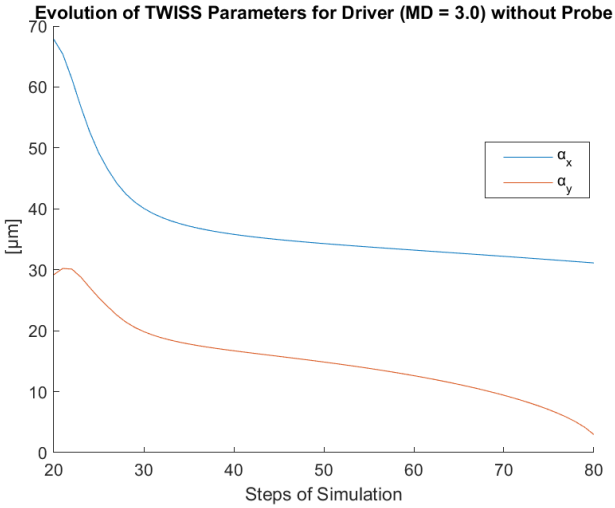
5.2. Difference plots

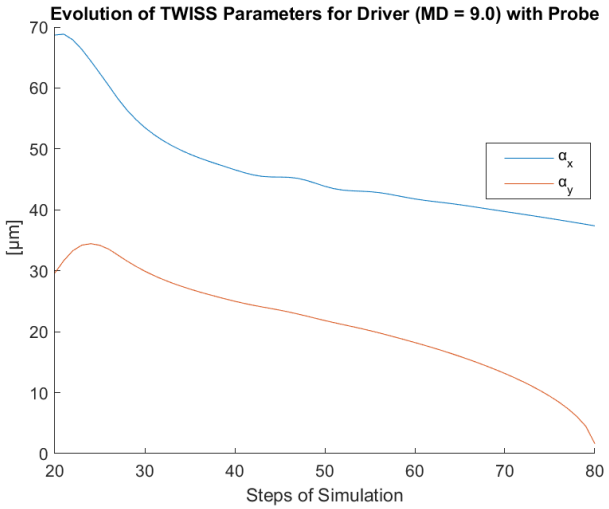
5.3. Density dependence of peak deviation


Evolution of TWISS Parameters for Driver (MD = 0.5) with Probe
[μm]
Steps of Simulation
α_x
α_y


Evolution of TWISS Parameters for Driver (MD = 0.5) without Probe
[μm]
Steps of Simulation
α_x
α_y


Evolution of TWISS Parameters for Driver (MD = 3.0) with Probe
[μm]
Steps of Simulation
α_x
α_y


Evolution of TWISS Parameters for Driver (MD = 3.0) without Probe
[μm]
Steps of Simulation
α_x
α_y


Evolution of TWISS Parameters for Driver (MD = 9.0) with Probe
[μm]
Steps of Simulation
α_x
α_y

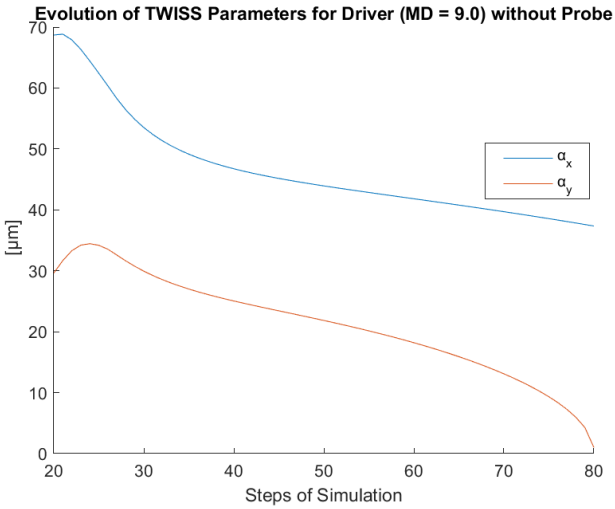
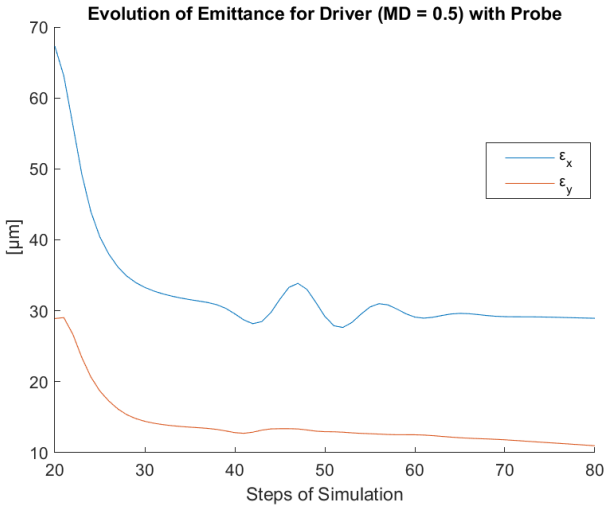
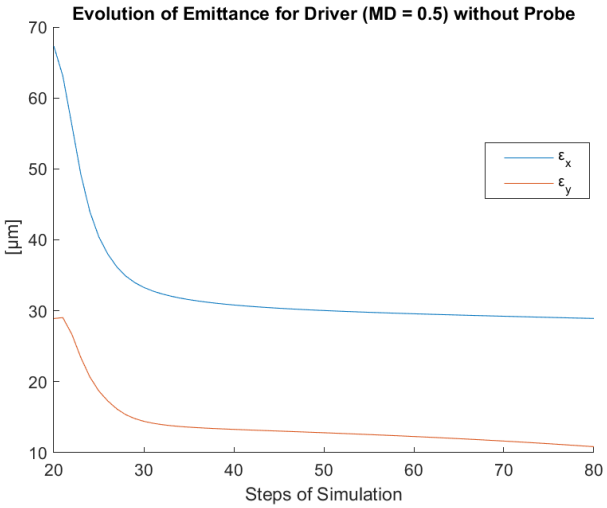
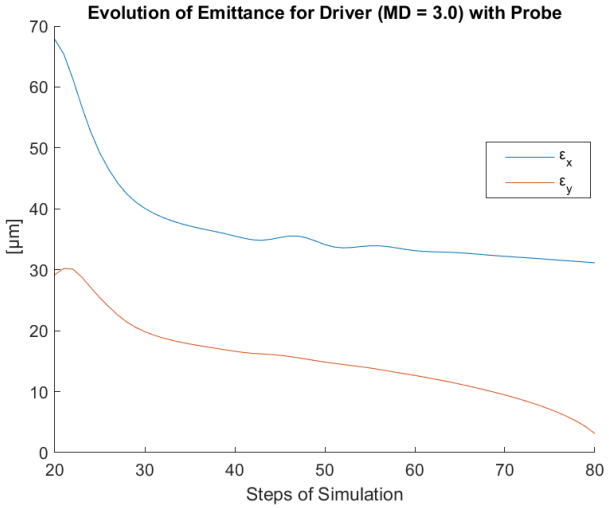
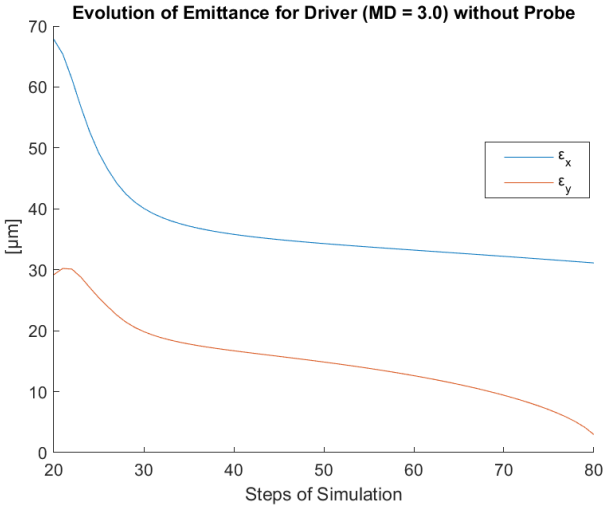

Evolution of TWISS Parameters for Driver (MD = 9.0) without Probe
[μm]
Steps of Simulation
α_x
α_y

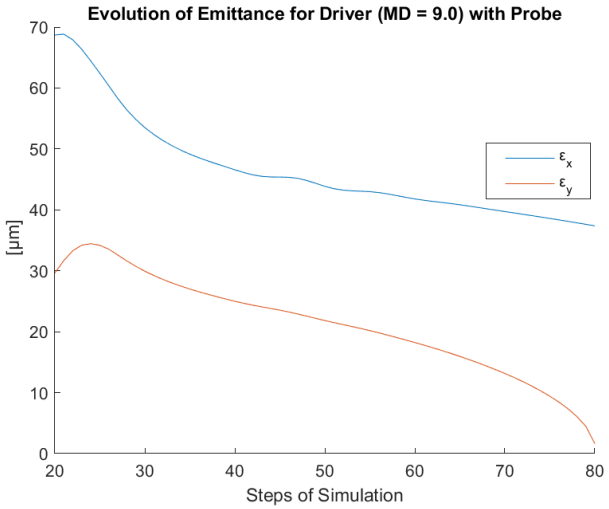
Figure 6: .


Evolution of Emittance for Driver (MD = 0.5) with Probe
[μm]
Steps of Simulation
ε_x
ε_y


Evolution of Emittance for Driver (MD = 0.5) without Probe
[μm]
Steps of Simulation
ε_x
ε_y


Evolution of Emittance for Driver (MD = 3.0) with Probe
[μm]
Steps of Simulation
ε_x
ε_y


Evolution of Emittance for Driver (MD = 3.0) without Probe
[μm]
Steps of Simulation
ε_x
ε_y


Evolution of Emittance for Driver (MD = 9.0) with Probe
[μm]
Steps of Simulation
ε_x
ε_y

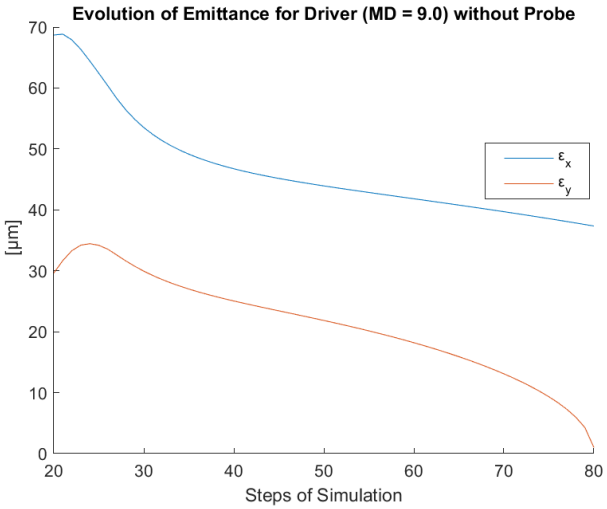

Evolution of Emittance for Driver (MD = 9.0) without Probe
[μm]
Steps of Simulation
ε_x
ε_y

Figure 7: .

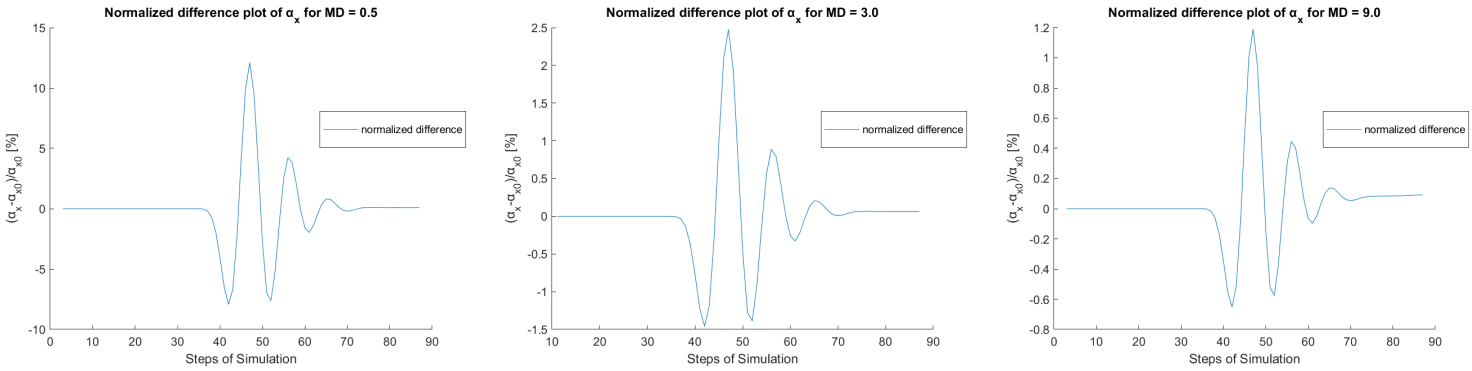


Figure 8: .

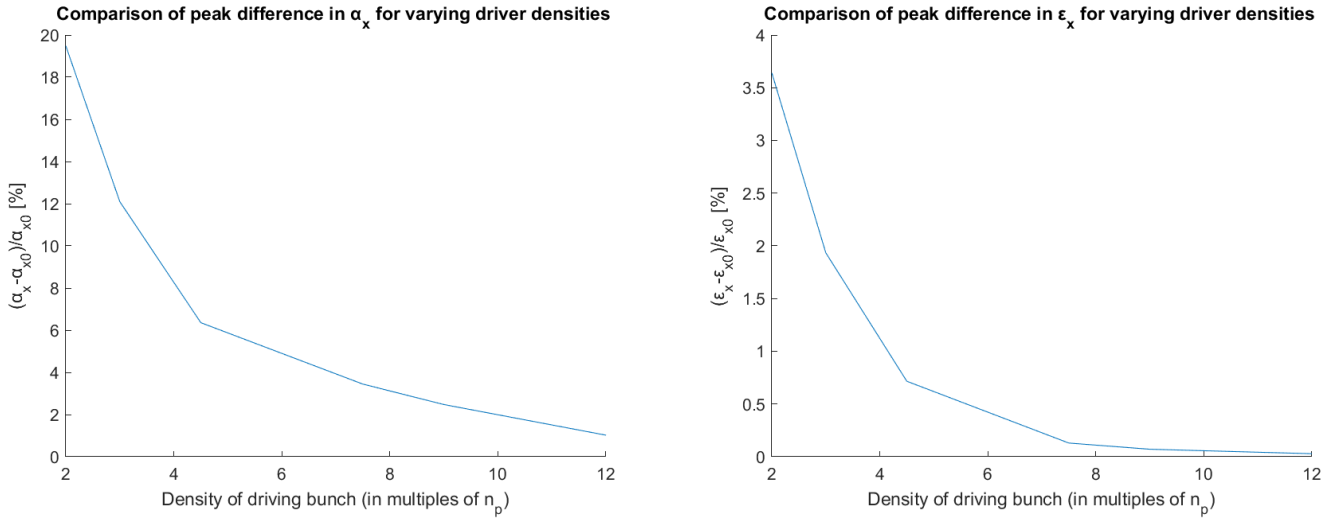


Figure 9: .

References

[1] Official DESY FLASH website *https://flash.desy.de/*

[2] Courtesy of *P. Niknejadi*

[3] Accelerator Beam Diagnostics, USPAS 2009 *U. Raich*

[4] Theoretical and Numerical Studies on the transport of transverse beam quality in plasma-based accelerators, PhD Thesis, Hamburg 2014 *T. Mehrling*

[5] Timon Mehrling's website *https://timonmehrling.com/research/pic-code-development/osiris/*

A.
OSIRIS Input file

```
!e-beam probe, 3D

simulation
{
  n0 = 3.0e18,
}

node_conf
{
  node_number(:) = 4, 4, 4,
  if_periodic(:) = .false., .false., .false.,
}

grid
{
  nx_p(:) = 512, 256, 256,
}

time_step
{
  dt = 0.069,
  ndump = 10,
}

restart
{
  ndump_fac = 0,
!  if_restart=.true.,
  if_remold=.true.,
}

space
{
  xmin(:) =  -55.0, -30., -20.,
  xmax(:) =  20.0, 20., 20.,
  if_move(:) = .true., .false., .false.,
}

time
{
  tmin = 0.0d0, tmax  = 60.0,
}

emf_bound
{
  type(:,1) = "vpml", "vpml",
  type(:,2) = "vpml", "vpml",
  type(:,3) = "vpml", "vpml",
}

diag_emf
{
  ndump_fac = 1,
  reports = "e1", "e2", "e3", "psi",
}

particles
```

```
{
  num_species = 3,
  low_jay_roundoff = .true.,
  ndump_fac = 1,
}

!=========Driving beam==========
species
{
  name = "driver",
  rqm= -1.0,
  num_par_x(:) = 1, 1, 1,
}

udist
{
  uth_type = "none",
  ufl(:) = 1000., 0., 0.,
  n_accelerate = 100,
}

profile
{
  density = 3.0,
  profile_type = "gaussian", "gaussian", "gaussian",
  gauss_center(:) = -22.0, -6.0, 0.,
  gauss_sigma(:) = 0.5, 1.0, 0.5,
  gauss_range(1:2,1) = -23.5, -20.5,
  gauss_range(1:2,2) = -9.0, -3.0,
  gauss_range(1:2,3) = -1.5,  1.5,
}

spe_bound
{
  type(:,1) = "open", "open",
  type(:,2) = "open", "open",
  type(:,3) = "open", "open",
}

diag_species
{
  ndump_fac = 1,
  ndump_fac_raw = 1,
  reports = "charge", "j1",
  ndump_fac_pha = 1,
  ps_xmin(:) = -55.0, -30., -20.,
  ps_xmax(:) =  20.,  20.,  20.,
  if_ps_p_auto = .true., .true., .true.,
  ps_nx(:) = 256, 256, 256,
  ps_np(:) = 256, 256, 256,
  phasespaces = "p1x1", "p2x2", "p3x3",
}

!=========Probe beam==========
species
{
  name = "Probe",
```

```
  rqm= -1.0,
  num_par_x(:) = 2, 2, 2,
}

udist
{
  uth_type = "none",
  ufl(:) = 0., 50., 0.,
  n_accelerate = 120,
}

profile
{
  density = 0.05,
  profile_type = "gaussian", "gaussian", "gaussian",
  gauss_center(:) = 9., -27.0, 0.,
  gauss_sigma(:) = 7.0, 1.0, 1.0,
  gauss_range(1:2,1) = -12., 30.,
  gauss_range(1:2,2) = -30., -24.,
  gauss_range(1:2,3) = -3.0, 3.0,
}

spe_bound
{
  type(:,1) = "open", "open",
  type(:,2) = "open", "open",
  type(:,3) = "open", "open",
}

diag_species
{
  ndump_fac = 1,
  ndump_fac_raw = 1,
  reports = "charge", "j1",
  ndump_fac_pha = 1,
  ps_xmin(:) =  -55.0, -30., -20.,
  ps_xmax(:) =  20.,  20.,  20.,
  if_ps_p_auto = .true., .true., .true.,
  ps_nx(:) = 256, 256, 256,
  ps_np(:) = 256, 256, 256,
  phasespaces = "p1x1", "p2x2", "p3x3", "x1x3",
}

!=========Backgroud Plasma==========
species
{
  name = "e",
  rqm=-1.0,
  num_par_x(:) = 2, 2, 2,
}

udist
{
  uth_type = "none",
  !uth(:) = .005, .005, .005,
}
```

```
profile
{
  density = 1.0,
  num_x = 6,
  x(:,1) = -1.d6,  -9.,  -4., 65., 70., 1.d8,
  fx(:,1) =   0.,   0.,   1.,  1.,  0.,   0.,
  x(:,2) = -1.d6, -20., -15.,  5., 10., 1.d8,
  fx(:,2) =   0.,   0.,   1.,  1.,  0.,   0.,
  x(:,3) = -1.d6, -20., -15., 15., 20., 1.d8,
  fx(:,3) =   0.,   0.,   1.,  1.,  0.,   0.,
}

spe_bound
{
  type(:,1) = "thermal", "thermal",
  type(:,2) = "thermal", "thermal",
  type(:,3) = "thermal", "thermal",
  uth_bnd(:,1,2) = 0.01, 0.01, 0.01,
  uth_bnd(:,2,2) = 0.01, 0.01, 0.01,
  uth_bnd(:,1,3) = 0.01, 0.01, 0.01,
  uth_bnd(:,2,3) = 0.01, 0.01, 0.01,
}

diag_species
{
  ndump_fac = 1,
  reports = "charge",
}
```

B.

Current FLASH parameters

FLASH Parameters 2017/18.
The unit for brilliance is B=photons/s/mrad^2/mm^2/0.1%bw. Note, that exact values depend on various conditions. Not all combinations within the parameter ranges given are possible.

Parameter	FLASH1	FLASH2
Electron beam energy	0.35 - 1.25 GeV	0.4 - 1.25 GeV
Normalised emittance at 1 nC (rms)	1.4 mm mrad	1.4 mm mrad
Energy spread	200 keV	500 keV
Electron bunch charge	0.1 - 1.2 nC	0.02 - 1 nC
Peak current	1 - 2.5 kA	1 - 2.5 kA
Electron bunches per second (typ./max)	300 / 5000	300 / 5000
Photon energy (fundamental)	24 - 295 eV	14 - 310 eV
Photon wavelength (fundamental)	51 - 4.2 nm	90 - 4 nm
Photon pulse duration (FWHM)	<30 - 200 fs	<10 - 200 fs
Peak Power (from av.)	1 - 5 GW	1 - 5 GW
Single photon pulse energy (average)	1 - 500 µJ	1 - 1000 µJ
Spectral Width (FWHM)	0.7 - 2 %	0.5 - 2 %
Photons per Pulse	10^{11} - 10^{14}	10^{11} - 10^{14}
Peak Brilliance	10^{28} - 10^{31} B	10^{28} - 10^{31} B

Osiris Raw Bunch Survey (Screen, Analyze, View, Write)

Input and Physical Constants:

```
sim = 'MD30_ESD';                    % remember to change plot tiles according to input file
path = strcat(sim, '/RAW/driver/');
ndata=87;

%plasma density
n_e=3e+24; % m^-3
%constants
c_light = 299792458; % m/s
q_electron = 1.602e-19; % C
m_electron = 9.109e-31; % kg
epsilon_0 = 8.85418782e-12; % F/m
e_E0 = 0.51099893; % MeV
one_over_4piepsilon_0 = 9e9; % N*m^2/C^2
omega_p = sqrt(n_e * q_electron.^2 / m_electron / epsilon_0)
```

```
omega_p = 9.7704e+13
```

Read H5 files and Make output files:

```
for i=1:ndata
    % Update of current filename
    filenum = num2str(i);
    if i < 10
        file = strcat(path, 'RAW-driver-00000', filenum, '.h5');
    else
        file = strcat(path, 'RAW-driver-0000', filenum, '.h5');
    end

    % read momentum, energy, position and charge
    p1 = hdf5read(file,'/p1');
    p2 = hdf5read(file,'/p2');
    p3 = hdf5read(file,'/p3');
    ene = hdf5read(file,'/ene');
    x1 = hdf5read(file,'/x1');
    x2 = hdf5read(file, '/x2');
    x3 = hdf5read(file, '/x3');
    q = hdf5read(file,'/q');
    %tg = hdf5read(file,'/tag');
```

Finding the Beam position and momentum:

Normalize Charge and find the total charge of the bunch. Then find the dimension of the bunch.

```
    q=abs(q);
    normalized_q = q/sum(q);
    bunch_q=sum(q);

    %% Position
    meanx(1,i)= dot(x1,normalized_q); % c/omega_p
    meanx(2,i)= dot(x2,normalized_q); % c/omega_p
```

```
    meanx(3,i)= dot(x3,normalized_q); % c/omega_p
    varx(1,i) = var(x1,normalized_q); % c/omega_p
    varx(2,i) = var(x2,normalized_q); % c/omega_p
    varx(3,i) = var(x3,normalized_q); % c/omega_p
    stdx(1,i) = sqrt(varx(1,i)); % c/omega_p
    stdx(2,i) = sqrt(varx(2,i)); % c/omega_p
    stdx(3,i) = sqrt(varx(3,i)); % c/omega_p


    %% Momentum
    meanp(1,i)= dot(p1,normalized_q); % m_e*c
    meanp(2,i)= dot(p2,normalized_q); % m_e*c
    meanp(3,i)= dot(p3,normalized_q); % m_e*c
    varp(1,i) = var(p1,normalized_q); % m_e*c
    varp(2,i) = var(p2,normalized_q); % m_e*c
    varp(3,i) = var(p3,normalized_q); % m_e*c
    stdp(1,i) = sqrt(varp(1,i)); % m_e*c
    stdp(2,i) = sqrt(varp(2,i)); % m_e*c
    stdp(3,i) = sqrt(varp(3,i)); % m_e*c

    %% Energy
    meanene(i) = dot(ene, normalized_q);
    varene(i) = var(ene,normalized_q);
    stdene(i) = sqrt(varene(i));
```

Ellipse Parameters (Twiss Parameters) and Energy Spread

```
    %% Emittance
    e_x(i) = sqrt(varx(2,i)*varp(2,i)-(dot([(x2-meanx(2,i)).*(p2-meanp(2,i))], normalized_q)).^2);
    e_y(i) = sqrt(varx(3,i)*varp(3,i)-(dot([(x3-meanx(3,i)).*(p3-meanp(3,i))], normalized_q)).^2);

    %% TWISS-Parameters
    beta_x(i) = varx(2,i)./e_x(i);
    beta_y(i) = varx(3,i)./e_y(i);
    gamma_x(i) = varp(2,i)./e_x(i);
    gamma_y(i) = varp(3,i)./e_x(i);

    alpha_x(i) = sqrt(beta_x(i)*gamma_x(i)-1);
    alpha_y(i) = sqrt(beta_y(i)*gamma_y(i)-1);

    %% Energy Spread
    enespr(i) = stdene(i)./meanene(i);


end
```

Output of Figures (Change caption according to input file)

```
%xwidth(9) = mean(stdx(2,:))*c_light./omega_p;
%ywidth(9) = mean(stdx(3,:))*c_light./omega_p;
%zwidth(9) = mean(stdx(1,:))*c_light./omega_p;



% figure, hold on
% title('Standard deviation of x and y for Driver with Probe')
% xlabel({'Steps of Simulation'}); ylabel({'Standard deviation [µm]'});
% plot(e_x(:)*c_light/omega_p*10e6)
% plot(e_y(:)*c_light/omega_p*10e6)
```

```
% legend('STD_x','STD_y','Position',[0.75 0.6 0.15 0.10])
% hold off

% BETA
f1 = figure, hold on
title('Evolution of TWISS Parameters for Driver (MD = 3.0) with Probe') %%% ATTENTION
xlabel({'Steps of Simulation'}); ylabel({' [μm]'});
plot([20:80], beta_x(20:80)*c_light/omega_p*10e6)
plot([20:80], beta_y(20:80)*c_light/omega_p*10e6)
legend('β_x','β_y','Position',[0.75 0.6 0.15 0.10])
hold off
savefig(strcat(sim, '/', sim, '_beta.fig'))
saveas(f1, strcat(sim, '/', sim, '_beta.png'))

%ALPHA
f2 = figure, hold on
title('Evolution of TWISS Parameters for Driver (MD = 3.0) with Probe') %%% ATTENTION
xlabel({'Steps of Simulation'}); ylabel({' [μm]'});
plot([20:80], alpha_x(20:80)*c_light/omega_p*10e6)
plot([20:80], alpha_y(20:80)*c_light/omega_p*10e6)
legend('α_x','α_y','Position',[0.75 0.6 0.15 0.10])
hold off
savefig(strcat(sim, '/', sim, '_alpha.fig'))
saveas(f2, strcat(sim, '/', sim, '_alpha.png'))

%EMITTANCE
f3 = figure, hold on
title('Evolution of Emittance for Driver (MD = 3.0) with Probe') %%% ATTENTION
xlabel({'Steps of Simulation'}); ylabel({' [μm]'});
plot([20:80], alpha_x(20:80)*c_light/omega_p*10e6)
plot([20:80], alpha_y(20:80)*c_light/omega_p*10e6)
legend('ε_x','ε_y','Position',[0.75 0.6 0.15 0.10])
hold off
savefig(strcat(sim, '/', sim, '_emittance.fig'))
saveas(f3, strcat(sim, '/', sim, '_emittance.png'))

%ENERGY SPREAD
```

Save file for comparison

```
% Driver with Probe
save(strcat(sim, '/', sim, '_data.mat'), 'alpha_x','alpha_y','beta_x','beta_y','e_x','e_y')
```

```
% Subject of comparison
ssim = 'MD75_ESD';
sfile = strcat(ssim, '/', ssim, '_data.mat');
```

```
sfile = 'MD75_ESD/MD75_ESD_data.mat'
```

```
sfile_wo = strcat(ssim, '_woP', '/', ssim, '_woP', '_data.mat');
```

```
sfile_wo = 'MD75_ESD_woP/MD75_ESD_woP_data.mat'
```

```
ax_wo = cell2mat(struct2cell(load(sfile_wo,'alpha_x')));
ax = cell2mat(struct2cell(load(sfile, 'alpha_x')));
bx_wo = cell2mat(struct2cell(load(sfile_wo,'beta_x')));
bx = cell2mat(struct2cell(load(sfile, 'beta_x')));
ex_wo = cell2mat(struct2cell(load(sfile_wo,'e_x')));
ex = cell2mat(struct2cell(load(sfile, 'e_x')));

ay_wo = cell2mat(struct2cell(load(sfile_wo,'alpha_y')));
ay = cell2mat(struct2cell(load(sfile, 'alpha_y')));
by_wo = cell2mat(struct2cell(load(sfile_wo,'beta_y')));
by = cell2mat(struct2cell(load(sfile, 'beta_y')));
ey_wo = cell2mat(struct2cell(load(sfile_wo,'e_y')));
ey = cell2mat(struct2cell(load(sfile, 'e_y')));
```

Difference Plots:

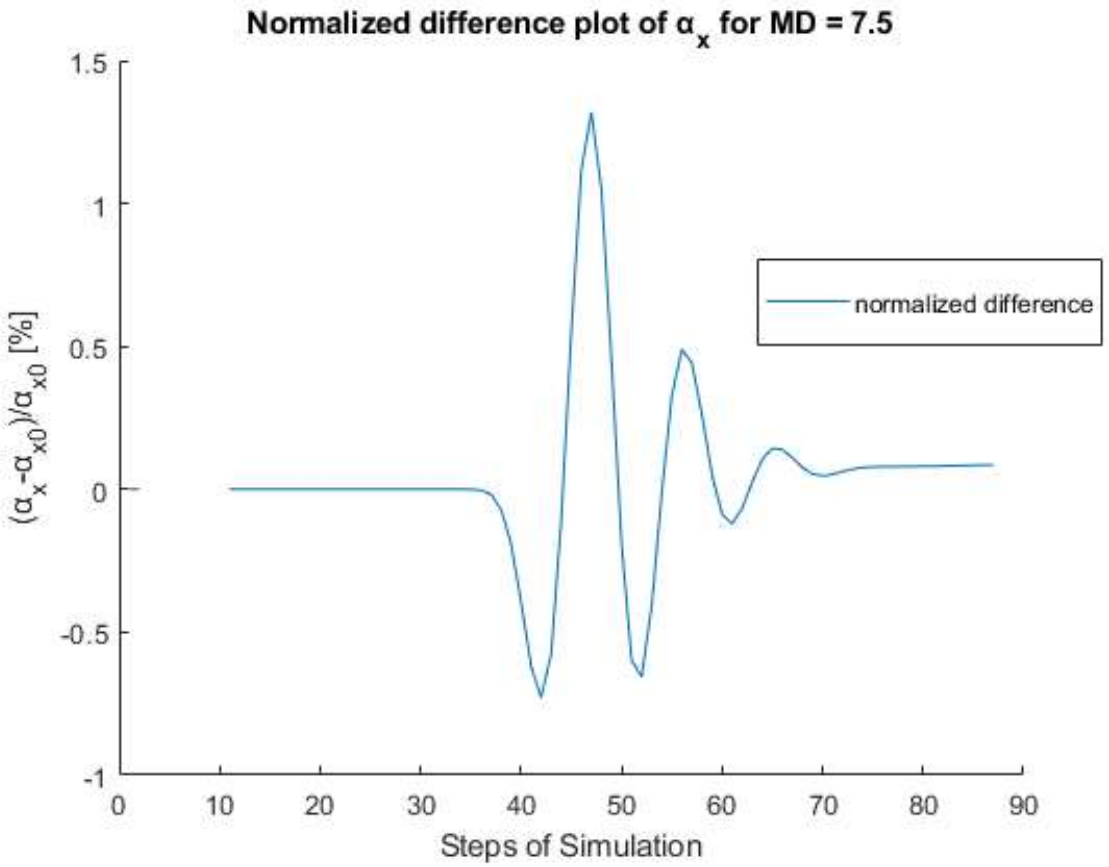
```
sf1 = figure, hold on
```

```
sf1 =
  Figure (4) with properties:

      Number: 4
        Name: ''
       Color: [0.9400 0.9400 0.9400]
    Position: [680 558 560 420]
       Units: 'pixels'

  Show all properties
```

```
title('Normalized difference plot of α_x for MD = 7.5') %%% ATTENTION
xlabel({'Steps of Simulation'}); ylabel({'(α_x-α_{x0})/α_{x0} [%]'});
plot(100.*(ax-ax_wo)./ax_wo)
legend('normalized difference','Position',[0.75 0.6 0.15 0.10])
hold off
savefig(strcat(ssim, '_alphaXdiff.fig'))
saveas(sf1, strcat(ssim, '_alphaXdiff.png'))
```



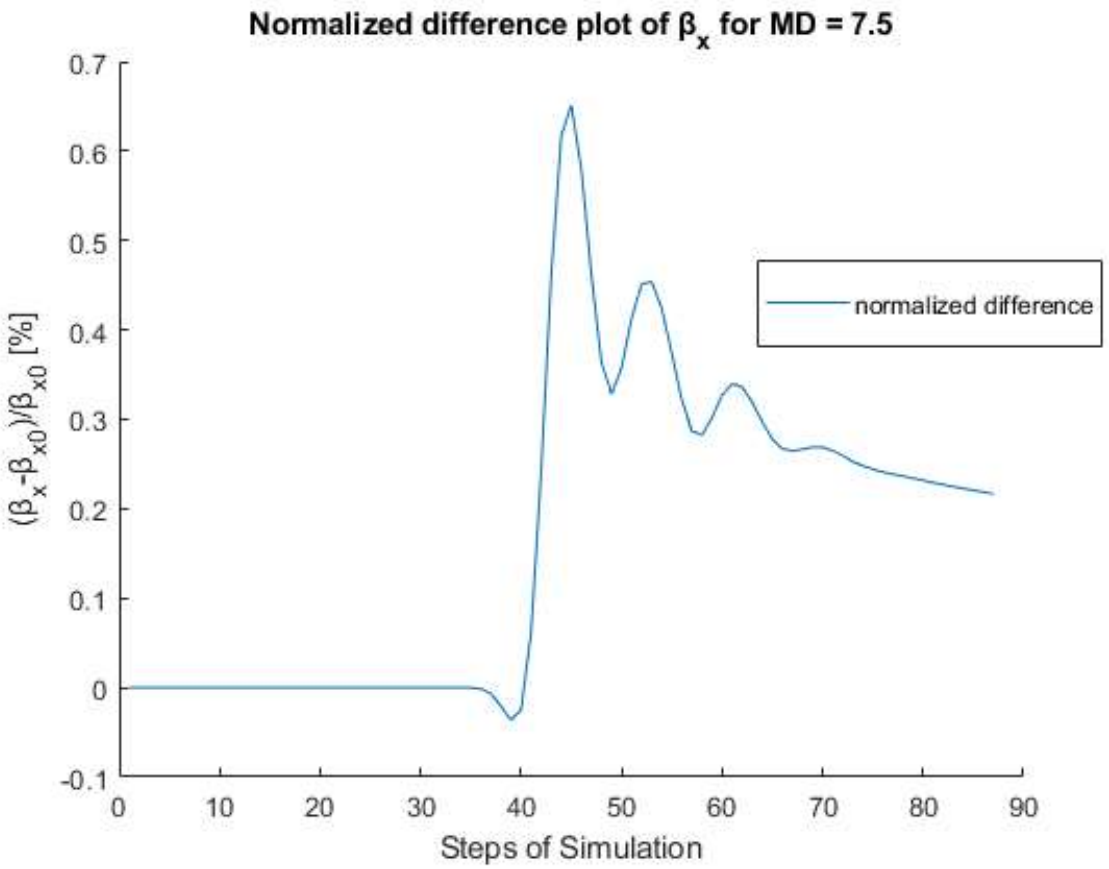

```
sf2 = figure, hold on
```

```
sf2 =
  Figure (5) with properties:

      Number: 5
        Name: ''
       Color: [0.9400 0.9400 0.9400]
    Position: [680 558 560 420]
       Units: 'pixels'

  Show all properties
```

```
title('Normalized difference plot of β_x for MD = 7.5') %%% ATTENTION
xlabel({'Steps of Simulation'}); ylabel({'(β_x-β_{x0})/β_{x0} [%]'});
plot(100.*(bx-bx_wo)./bx_wo)
legend('normalized difference','Position',[0.75 0.6 0.15 0.10])
hold off
savefig(strcat(ssim, '_betaXdiff.fig'))
saveas(sf2, strcat(ssim, '_betaXdiff.png'))
```



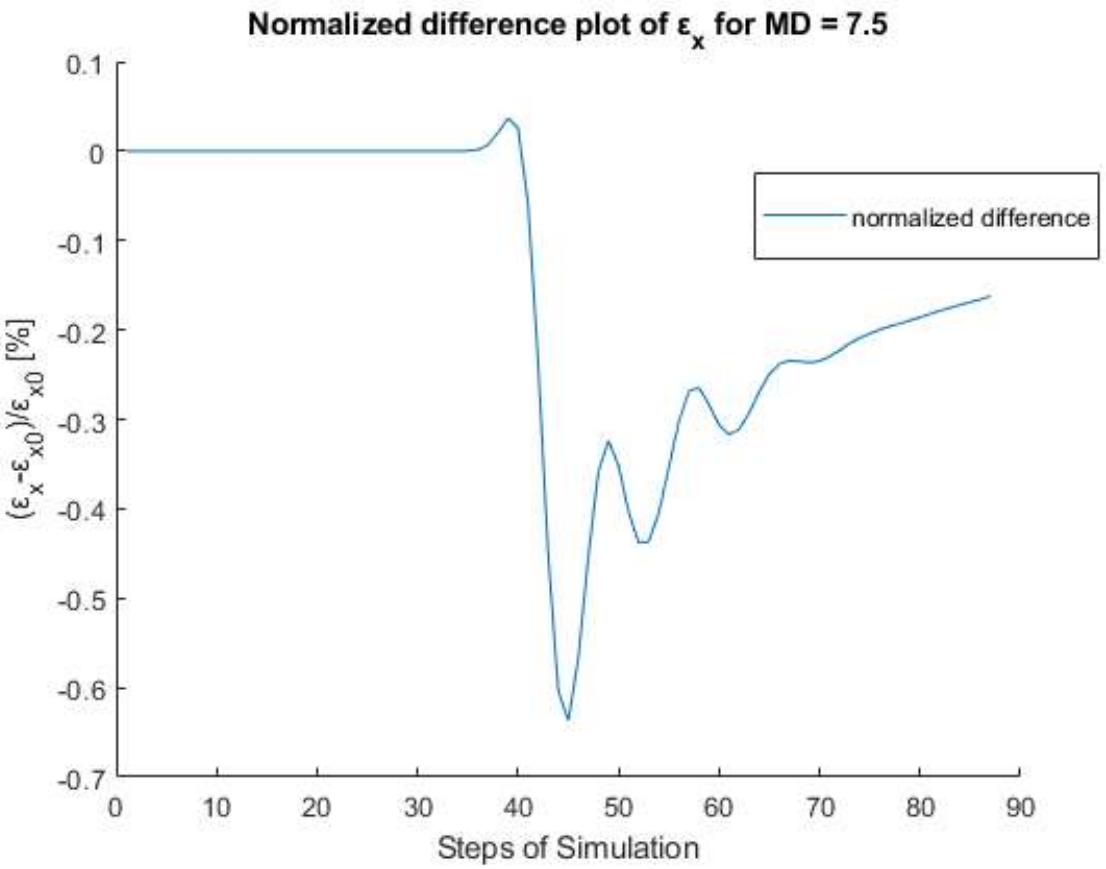

```
sf3 = figure, hold on
```

```
sf3 =
  Figure (6) with properties:

      Number: 6
        Name: ''
       Color: [0.9400 0.9400 0.9400]
    Position: [680 558 560 420]
       Units: 'pixels'

  Show all properties
```

```
title('Normalized difference plot of ε_x for MD = 7.5') %%% ATTENTION
xlabel({'Steps of Simulation'}); ylabel({'(ε_x-ε_{x0})/ε_{x0} [%]'});
plot(100.*(ex-ex_wo)./ex_wo)
legend('normalized difference','Position',[0.75 0.7 0.15 0.10])
hold off
savefig(strcat(ssim, '_emittXdiff.fig'))
saveas(sf3, strcat(ssim, '_emittXdiff.png'))
```




```
% plot((ay-ay_wo)./ay_wo)
% plot((by-by_wo)./by_wo)
% plot((ey-ey_wo)./ey_wo)
```