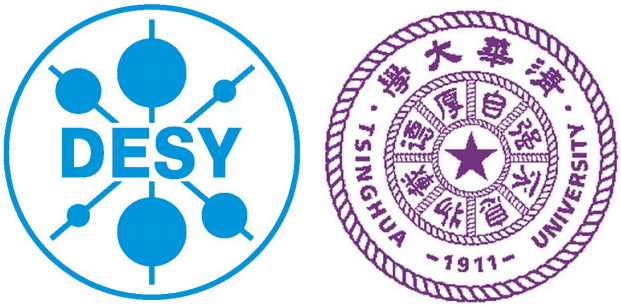

Three-Dimensionally Visualized Project Based on IDL Program in Beam-Driven Wakefield Acceleration

Dezhi Cao

Tsinghua University, China

Supervisor

Timon Mehrling

FLA Group

DESY Summer Student Program, 2015

September 4, 2015

Abstract

This report describes project about three-dimensional visualization based on IDL(Interactive Data Language) used for results produced by simulation in HDF5 format. Reading data file and building graphical models by using IDL's library, we can create pictures in png format and export movie in mp4 format. Written scripts have been debuged in IDLDE(IDL Development Environment) so that we can run in command line directly with adjustable parameters.

The work will help us to understand physical procedure of beam-driven wakefield acceleration in space and time scale,and that is what I have done in FLA(Forschung Linear Accelerator) group during the DESY Summer Student Program.

Contents

1 Introduction

1.1 Plasma Wakefield Acceleration

Plasma wakefield acceleration of electrons with high-gradient field, especially in laser-driven wakefield accelerators and beam-driven wakefield accelerators, has progressed rapidly, leading to an improvement in the quality of the accelerated beams.

People has validated laser-driven wakefield accelerators(LWFA) as a promising techniques for future because of some realization such as GeV-beam [1], and the application of the generated beams to drive compact extreme ultraviolet [2] and x-ray sources [3]. At the same time, beam-driven plasma wakefield acceleration(PWFA) made great progress culminating in the demonstration of energy doubling of part of the 42 GeV SLAC electron beam [4].

However,insufficient control over the electron-injection process in PWFA have a great influence on the quality of electron bunches. That's why theoretic simulation and experiments of evolution of electron beam in PWFA process is extremely significant. Therefore, three dimensional visualization used for analyzed data will enhance our understanding of phenomenon in some key points, such as hosing instability.

1.2 PIC Simulation and Osiris Code

The PIC(Particle-In-Cell) technique is an excellent approach to simulate a huge variety of plasma-particles-behaviours, especially highly nonlinear and kinetic processes that occur during high-intensity particle and laser beam-plasma interactions.It's able to compute large sets of position, momentum, energy and their self-consistent fields of particles [5].

Osiris, a three-dimensional, relativistic, massively parallel, object oriented particle-in-cell code for modeling plasma based accelerators, was developed in Fortran 90 and then runs on multiple platforms (Cray T3E, IBM SP, Mac clusters). It is through an object-oriented programming style that divides the code and data structures into independent classes of objects. This programming style maximizes code reusability, reliability, and portability, which break up the large problem of a simulation into a set of essentially independent smaller problems that can be solved separately from each other.

Osiris code achieved this by handling different aspects of the problem in different modules (classes) that communicate through well-defined interfaces. That's the reason we discuss the object-oriented design of the code, the encapsulation of system dependent code and the parallelization of the algorithms involved. It can make it possible allowing individuals in a code development team to work independently. Therefore,implementation of communications as a boundary condition problem and other key characteristics of the code, such as the moving window, open-space and thermal bath boundaries, arbitrary domain decomposition, 2D (cartesian and cylindric) and 3D simulation modes, electron sub-cycling, energy conservation and particle and field diagnostics are always discussed [6].

Finally the results from three-dimensional simulations of particle and laser wake-

field accelerators are presented, in connection with the data analysis and visualization infrastructure developed to post-process the scalar and vector results from PIC simulations [7].

1.3 IDL Introduction and Code Style

IDL, short for Interactive Data Language, is a programming language used for data analysis. It is popular in particular areas of science, such as astronomy, atmospheric physics and medical imaging. IDL shares a common syntax with PV-Wave and originated from the same codebase, though the languages have subsequently diverged in detail.

IDL is vectorized, numerical, and interactive, and is commonly used for interactive processing of large amounts of data (including image processing). The syntax includes many constructs from Fortran and some from C.

IDL has been applied widely in space science, for example in solar physics. The European Space Agency used IDL to process almost all of the pictures of Halley's Comet taken by the Giotto spacecraft. The team repairing the Hubble Space Telescope used IDL to help them diagnose anomalies in the main mirror. In 1995, astronauts on board a space shuttle used IDL loaded on a laptop to study ultraviolet radiation. Currently, amongst other applications, IDL is being used for most of the analysis of the SECCHI part of the STEREO mission at NRL, USA, and at the Rutherford Appleton Laboratory, UK.

Although IDL has widely used in scientific field. Actually, facing with nearly 60 folders and 500 code file is a big challenge especially it is totally new to me. The first thing I can do is studying the feature of IDL language. Spending some time on learning it, I make a nest about its organization of code below(Fig.1) in my opinion. After comparing to C and Matlab I have learned before, it has different regulations such as definition of function and procedure, transferring parameters between code files, IO rules and etc. But it is also the high level language in deeply thinking.

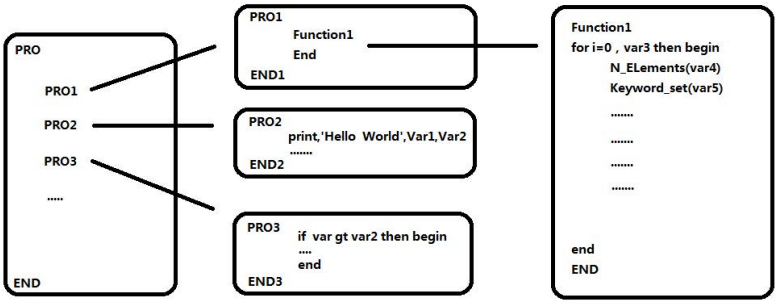

Figure 1: Example of Organized Code Structure

In addition, I provide some code examples in scripts of IDL shown in Fig.2.

```
PRO Osiris_Analysis_3D, _EXTRA=extrakeys, $
                        ; Axis Ranges
                        XRANGE = xrange, YRANGE = yrange, ZRANGE = zrange, $
                        ; Titles and Labels
                        XTITLE = XAxisTitle, YTITLE = YAxisTitle, ZTITLE = ZAxisTitle, $
                        TITLE = PlotTitle, SUBTITLE = PlotSubTitle,  TIME_UNITS = time_units, $
                        ; Slicer Operation
                        SLICER = use_slicer, $
                        EXTRAMODEL = extraModel,$
                        ;File
                        FILENAME = FileName, $       ; (in) name of the file to open
                        PATH = filepath, $         ; (in) path to the file to open
                        N_ISOLEVEL = n_Isolevel,$   ;(in)number of iso-surface,
                        ;Output parament
                        OUTNAME = outname, $        ; (out) name to the file to output
                        OUTFORMAT = outformat ,$    ; (out) format to the file to output
                        OUTPATH = outpath           ; (out) path to the file to output

; ********************************************************************************************** Parameters

; TIME_UNITS
;
; Units to display next to the time information

if N_Elements(time_units) eq 0 then time_units = '1 / !Mw!Dp!N'


; ********************************************************************************************** Main Code

; Gets the Data


Osiris_Open_Data, _EXTRA=extrakeys, pData, $
                  TIMEPHYS = time, $
                  N = dxN, FORCEDIMS = 3, $
                  XRANGE = xrange, YRANGE = yrange, ZRANGE = zrange, $
                  XAXIS = XAxisData, YAXIS = YAxisData, ZAXIS = ZAxisData, $
                  DATATITLE = DataName, DATALABEL = DataLabel, DATAUNITS = Units, $
                  XLABEL = x1label, XUNITS = x1unit, $
                  YLABEL = x2label, YUNITS = x2unit, $
                  ZLABEL = x3label, ZUNITS = x3unit, $
                  FILENAME = FileName, $        ; (in) name of the file to open
                  PATH = filepath, $         ; (in) path to the file to open
```

Figure 2: Code Example of the Analysis File

2 Three Dimensional Visualization

2.1 Flow Chart of Project

The whole folder set consists of sixty nine sub-folders and five hundred and thirty three code files, but my task is drawing the three dimensional picture and output the movie. Therefore, running main code file such as "osiris_analysis_3d.pro", "osiris_movie_3d.pro", "osiris_particles.pro", "osiris_particles_movie.pro" and its callback function contained in folder "plot3","misc" and so on, are my targets during six weeks. Now I figured the relationship between main process and making two flowcharts here. The one is plotting picture for plasma surface(Fig. 3) and the other is drawing picture with particles models(Fig. 4).

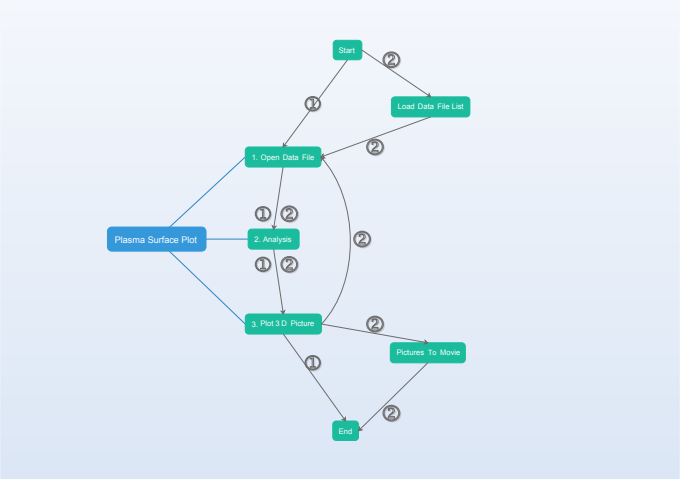

Figure 3: Flow Chart of Main Function in Plotting Plasma Surface

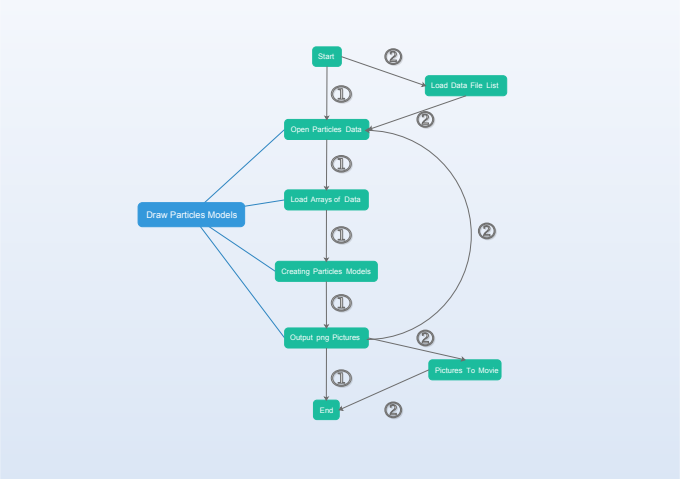

Figure 4: Flow Chart of Main Functions in Particles Modeling

2.2 Plotting Plasma Density Surface in Three Dimension

Taking notes for variables help us understand meaning of transferred parameter, then debugging the programmer step by step solve problems in functions that using wrong index of array. Thus, we can easily plot pictures of plasma surface shown in Fig. 5.

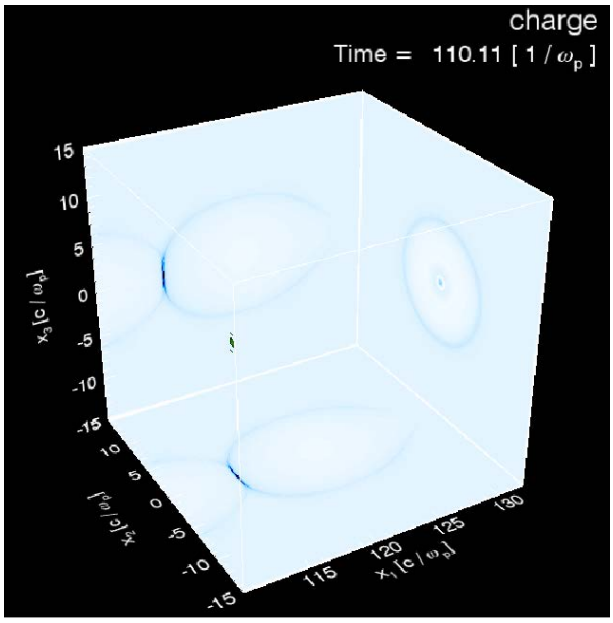

Figure 5: 3D Plot of Plasma Density After a Travel Time of $110.11[1/\omega_p]$

Optimizing parameters and choosing proper value to generate more plasma surfaces, and then drawing plasma surface in different color will output good-looking picture.

From the picture shown in Fig. 6, we can see the plasma density distribution in time scale $110.11[1/\omega_p]$. It shows us shape of plasma surface which looks like a bullet directly after electron beam goes through.

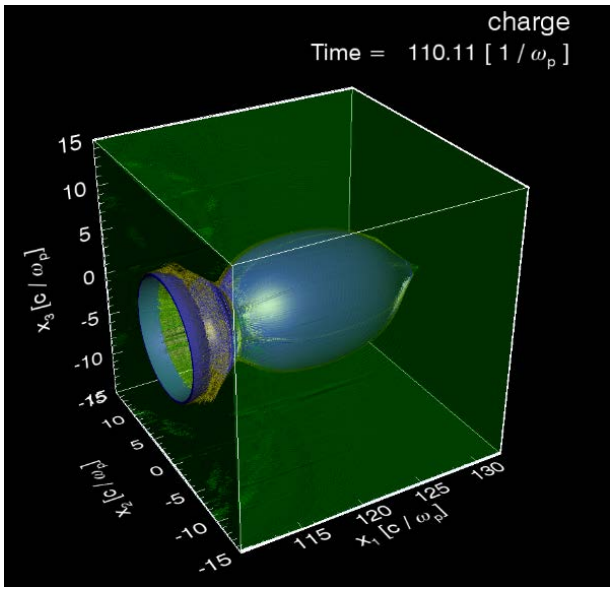

Figure 6: Optimized 3D Plot of Plasma Density After a Travel Time of $110.11[1/\omega_p]$

The distribution of longitudinal electric field is surrounded in electron beam. That's main driven force which accelerates the second electron beam. The picture is shown in Fig. 7

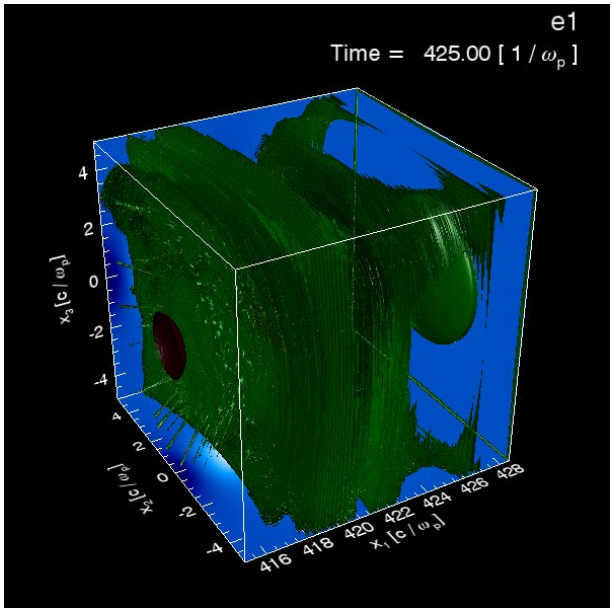

Figure 7: 3D plot of Longitudinal Electric Field After a Travel Time of $425.00[1/\omega_p]$

In different time scale, we can see evolution of plasma density ,such as in time scale $110.11[1/\omega_p]$, shown in Fig. 8. From that, we know that accelerated beam has becomed relativistic because its shape is narrow than before.

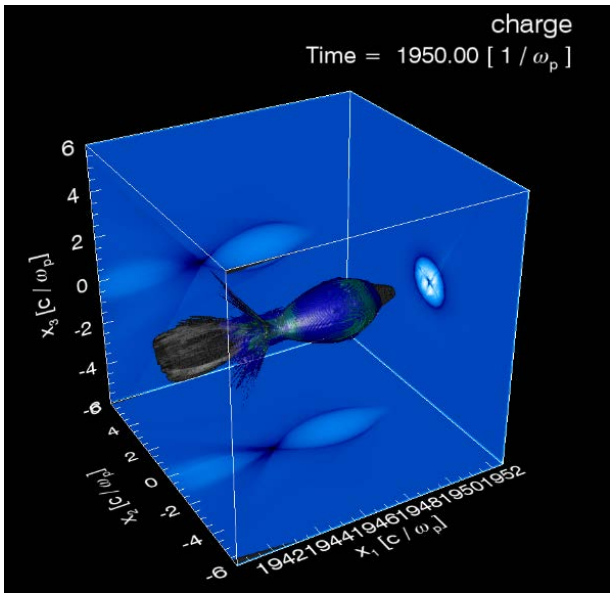

Figure 8: Optimized 3D Plot of Plasma Density After a Travel Time of $1950.00[1/\omega_p]$

2.3 Plotting Three-Dimensional Models of Electron Beam Particles

In addition, properties of particles, including position, momentum, energy and etc, simulated by osiris code, will be loaded into variables through the pointer pSpecies according to their string labels “x1”, “x2”, “x3”, “p1”, “p2”, “p3”, “ene”, “q”, “tag” (Fig. 9).

```
142   Osiris_Open_Particles, FILE = file, NUMPAR = numpar, NUMSPECIES = numspecies, $
143                          SPIDX = spidx, SPNPAR = spnpar, TIME = time, ITER = iter, $
144                          NODE = node, $
145                          pSpecies
146
147   ;To avoid some error when particles is zero;
148   if (spnpar eq 0)  then GOTO, SKIP
149
150    spID = 0
151    ;calulate gamma
152    gamma = sqrt(1+total(((*pSpecies[spID]).part[*].p[*])^2,1))
153
154    case x1dir of
155  'x1'    : xdata = (*pSpecies[spID]).part[*].x[0]
156  'x2'    : xdata = (*pSpecies[spID]).part[*].x[1]
157  'x3'    : xdata = (*pSpecies[spID]).part[*].x[2]
158  'p1'    : xdata = (*pSpecies[spID]).part[*].p[0]
159  'p2'    : xdata = (*pSpecies[spID]).part[*].p[1]
160  'p3'    : xdata = (*pSpecies[spID]).part[*].p[2]
161  'gamma': xdata = gamma
162  'q'     : xdata = (*pSpecies[spID]).part[*].q
163    else :begin
164          print,"Wrong input,Input parameter:X1DIR = ?('x1','x2','x3','p1','p2','p3','gamma','q')"
165          return
166          end
167    endcase
168
169    if (dim ge 2) then begin
170      case x2dir of
171        'x1'    : ydata = (*pSpecies[spID]).part[*].x[0]
172        'x2'    : ydata = (*pSpecies[spID]).part[*].x[1]
173        'x3'    : ydata = (*pSpecies[spID]).part[*].x[2]
174        'p1'    : ydata = (*pSpecies[spID]).part[*].p[0]
175        'p2'    : ydata = (*pSpecies[spID]).part[*].p[1]
176        'p3'    : ydata = (*pSpecies[spID]).part[*].p[2]
177        'gamma': ydata = gamma
178        'q'     : ydata = (*pSpecies[spID]).part[*].q
179        else:begin
180             print,"Wrong input,Input parameter:X2DIR = ?('x1','x2','x3','p1','p2','p3','gamma','q')"
181             return
182             end
183      endcase
184    end
```

Figure 9: Loading Particle Data from Pointer

However, overflowing the memory will be a problem if we want to create a lot of particle models from the simulation results. Setting a limitation of model numbers and generating random number to pick up particles sample from the whole set will solve this problems in a proper way(Fig. 10).

```
case (shape) of
  0: Sphere, vert, poly, quality, [0.,0.,0.], 1.0, ASPECTRATIO = sphereratio
  1: Cube, vert, poly, [0.,0.,0.], 1.0, ASPECTRATIO = sphereratio
  2: Cone, vert, poly, quality, [0.,0.,-0.5], [0.,0.,+0.5], 1.0, ASPECTRATIO = sphereratio
  3: Cylinder, vert, poly, quality, [0.,0.,-0.5], [0.,0.,+0.5], 1.0, ASPECTRATIO = sphereratio
endcase


rad = radius*LBoxMax
if (n_elements(rad) eq 1) then begin
  ; Same radius for all points
  ; use faster algorithm

  vert = vert * rad

  s = size(vert, /dimensions)
  nvert = s[1]
  vertp = fltarr(3, nvert)

  ;default number of plot point(slow & lots of particle)
      if (npoints lt 100000) then begin
        plot_point = npoints
      ;generate index number within polot_point
      selected_points = lindgen(npoints)
      end else begin
        if (npoints gt 100000) and (npoints lt 200000) then plot_point = floor(npoints/2.0)
        if (npoints gt 200000) then plot_point = 100000
      ;generate random number within polot_point
        selected_points = floor(randomu(seed,plot_point)*npoints)
      end

  for i=0L, plot_point-1 do begin
    vertp[0,*] = vert[0,*] + pos[0,selected_points[i]]
    vertp[1,*] = vert[1,*] + pos[1,selected_points[i]]
    vertp[2,*] = vert[2,*] + pos[2,selected_points[i]]

    oModel -> Add, obj_new('IDLgrPolygon',vertp, $
                           POLYGON = poly, STYLE = 2, Shading = 1, $
                           COLOR = reform(color_all[*,i]) , $
                           REJECT = 1, UVALUE = name)
  end
```

Figure 10: Generating Random Number and Creating Models

Finally we can draw particle models according to its position to show its movement in short time scale. It is shown in Fig. 11.

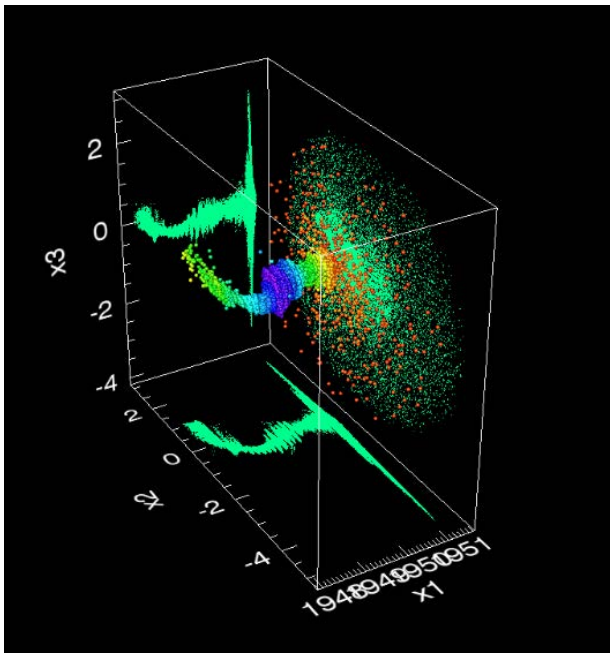

Figure 11: Plots of Self-Injection Electron Beam

2.4 Creating MP4 Movie

Adding loop that reading file list of data in directory, we can plot three dimensional picture continuously. Meanwhile, producing png format photographs and link them up in proper(8 fps) speed at the same directory. Finally we are able to create movie in mp4 format. The first movie shows kinetic behavior of evolution of plasma density(Fig. 12).

Figure 12: Evolution of Plasma Density(Click It to Play)

The second one is a movie of driven electron beam(Fig. 13).

Figure 13: Particles of Driven Electron Beam(Click It to Play)

The last one shows us how self-injection electron beam developes with time. From

that movie, we can understand the hosing instability at the end of electron beam through a lot of particle models. The movie is shown in Fig. 14.

Figure 14: Particles of Self-Injection Election Beam(Click It to Play)

3 Summary

After developing scripts of IDL and coping with simulation results, we can draw pictures of plasma surfaces and particle models in three dimension. And then playing movie in the frequency of eight picture per second to show dynamic movement of electron beam. It is obviously that we will get more clearly perspective about beam-driven wakefield acceleration in physical sight.

From the pictures and movies, we can change parameters of the simulation to get better results, leading to more specific and clear physical description in beam-driven plasma wakefield acceleration. It will be hopeful to find new physical principles in this fields.

4 Acknowledge

I really would like to thank my supervisors Timon Mehrling, Albertos Martinez de la Ossa, and Zhanghu Hu, and all the members of the DESY FLA Group for their help. Because of their patience and kindness with my work here and the opportunity they gave me to join their research group , I have an exciting experience in these two months.

I also have to thank the Summer Student Program organizers that made this great opportunity possible and made all their efforts, which is the most important reason we can take part in the project. Besides, thanks to Olaf for all the ideas and suggestions he gave us to enjoy our stay in Hamburg.

References

[1] Nature. Physics. 2, 696(2006). *W.P. Leemans, B.Nagler, A.J. Gonsalves, Cs. To'th, K.Nakamura, C.G.R. Geddes, E. Esarey, C.B. Schroeder, and S.M. Hooker*

[2] Nature. Physics. 5, 826 (2009). *Matthias Fuchs1,2, Raphael Weingartner1,2, Antonia Popp1, Zsuzsanna Major1,2, Stefan Becker2, Jens Osterhoff1,2, Isabella Cortrie2, Benno Zeitler2, Rainer H?rlein1,2, George D. Tsakiris1, Ulrich Schramm3, Tom P. Rowlands-Rees4, Simon M. Hooker4, Dietrich Habs1,2, Ferenc Krausz1,2, Stefan Karsch1,2, Florian Grner1,2*

[3] Nature. Physics. 6, 980 (2010). *S. Kneip, C. McGuffey, J.L. Martins, S.F. Martins, C. Bellei, V. Chvykov, F. Dollar, R. Fonseca, C. Huntington, G. Kalintchenko, A. Maksimchuk, S.P.D. Mangles, T. Matsuoka, S.R. Nagel, C.A.J. Palmer, J. Schreiber, K. Ta Phuoc, A.G.R. Thomas, V. Yanovsky, L.O. Silva, K. Krushelnick, Z. Najmudin*

[4] Nature (London) 445, 741 (2007). *Ian Blumenfeld1, Christopher E. Clayton2, Franz-Josef Decker1, Mark J. Hogan1, Chengkun Huang2, Rasmus Ischebeck1, Richard Iverson1, Chandrashekhar Joshi2, Thomas Katsouleas3, Neil Kirby1, Wei Lu2, Kenneth A. Marsh2, Warren B. Mori2, Patric Muggli3, Erdem Oz3, Robert H. Siemann1, Dieter Walz1, Miaomiao Zhou2*

[5] Web Page of FLA Group , *http://plasma-wiki.desy.de/PIC-simulation(permission limited)*

[6] Web Page of Osiris Documention, *https://osiris.ist.utl.pt/index.php/Osiris(permission limited)*

[7] Web Page of FLA Group ,*http://plasma-wiki.desy.de/PIC-simulation(permission limited)*

[8] IDL-Wikipedia *https://en.wikipedia.org/wiki/IDL-programming-language*