Appendix A

```
%Defining known parameters
lamb=0.001027; %laser wavelength(mm)
radi_ExPup=8.5; %aperture radius (mm)
w_ini=7.075; %beam width (mm)
Ein=3*10^-6; %laser pulse energy (J)
Fllin=2500; %titanium fluence threshold (J/m^2)
temps=450*10^-15; %laser pulse duration (s)
cn=0.00265; %speed of light multiplied by the electrical permittivity of vacuum (F/s^2)
Z_ini=-13; %focalization distance (mm)
tilt_deg=0.7; % lens tilt angle (with respect to the xy plane) (degrees)
angle1=26; % angle of rotation with respect to the x axis (degrees)
A=1; % electric field amplitude (V/m)
desp_openx=1.02; %pupil displacement in the x axis (mm) (real value*2.83)
desp_openy=-0.50; %pupil displacement in the y axis (mm) (real value*2.83)

%Gaussian beam parameters calculation
k=2*pi/lamb; %wave vector (rad/mm)
w0=w_ini/(sqrt(1+(pi*w_ini^2/lamb/z_ini)^2)); %beam waist width (mm)
LRayl=pi*w0^2/lamb; %Rayleigh range (mm)
R_ini=z_ini*(1+(LRayl/z_ini)^2); %curvature radius at the initial plane(mm)
gouy_ini=atan(z_ini/LRayl); % Gouy phase

%Defining the matrix dimensions and matrix positions
N=2048; iN=-N/2:(N/2-1); %number of points of the matrix
L_ini=20.9*radi_ExPup; %length of the matrix (mm)
T_ini=L_ini/N; %digitalization interval (mm)
[x,y]=meshgrid(iN*T_ini,iN*T_ini); %generating the matrix positions
x2= x.*cosd(angle1)-y.*sind(angle1); %changing the coordinates
y2= x.*sind(angle1)+y.*cosd(angle1);

%Calculating the electic values at the output of the objective
f_ini = A * exp(1i*(k*z_ini-gouy_ini)) * exp((1i*k/2/R_ini-
1/w_ini^2)*((x2*cosd(tilt_deg)).^2+y2.^2)); %the cosine implements the objective tilt
%The real distribution is implemented here

%Taking into account the optical path with tilt in x2 direction
tilt_x2 = exp(1i*k.*x2* sind(tilt_deg)/ cosd(tilt_deg));
f_ini = f_ini.* tilt_x2;

%The beam is cut by the pupil
pupil=sqrt((x-desp_openx).^2+(y-desp_openy).^2)< radi_ExPup;
f_ini=f_ini.*pupil;

%The electric field values are recalculated to match the laser pulse energy
E= sum(sum(abs(f_ini).*abs(f_ini)*T_ini*T_ini*10^-6*cn/2))*temps;
f_ini=f_ini*sqrt(Ein/E);

%Plot of the intensity distribution at the output of the objective
zoom=-400:399;
figure(1), imshow(abs(f_ini(zoom+N/2+1,zoom+N/2+1)).^2,[]),colormap(jet);

%Using the propagator to propagate to a certain distance
```

```
d=12.971;
[Uf Tf]=propaga_d(lamb,N,f_ini,T_ini,d);

%Plot of the intensity distribution  at certain position
figure(2), imshow(abs(Uf(zoom+N/2+1,zoom+N/2+1)).^2,[]),colormap(jet);
%Plot of the threshold image at certain position
figure(3),imshow((abs(Uf(zoom+N/2+1,zoom+N/2+1)).^2)<Fllin*2/(cn*temps),[]);
%Calculation of the beam with at certain position
beamwidth=sqrt(sum(sum((abs(Uf).^2)>(max(max(abs(Uf).^2))/exp(2))))*Tf*Tf/pi());
```

Appendix B

```
%read the real intensity distribution matrix before the objective
%first column= x position, second column= y position, third column = value
M=csvread('real_intensity_distribution.txt');
p=size(M);
NN=zeros(240,320);
M(p(1),2)
for r=1:p(1)
    NN(M(r,1)+1,M(r,2)+1)=M(r,3);
end
%normalize the matrix
 NN=NN/max(max(NN));
%substitute the simulated values for the real ones
zoomy=-119:118;
zoomx=-157:155;
f_ini(zoomy+N/2+1,zoomx+N/2+1)=sqrt(NN(1:238,8:320)).*exp(1i*angle(f_ini(zoomy+N/2+1
,zoomx+N/2+1)));
```

Appendix C

```
%propagator based on the Fresnel diffraction equations [5]
%return electric values and the digitalization  value
%the function acts using wavelength, number of matrix points, electric field, digitalization value
and propagation distance
function [Uf Tf]=propaga_d(lamb,N,Ui,Ti,d)
    iN=-N/2:(N/2-1);
    if d==0,        Uf=Ui; Tf=Ti; %it allows to not do propagation
    else
        k=2*pi/lamb;
        [x,y]=meshgrid(iN*Ti,iN*Ti);
        fase_i=exp(1i*k/2/d*(x.^2+y.^2));
        lamb*d/2/(N/2*Ti);     % digitalization  limit  value
        integ=fftshift(fft2(fftshift(Ui.*fase_i)))*Ti*Ti;
        Tf=lamb*d/N/Ti;  %new digitalization  values
        x=x*Tf/Ti;  y=y*Tf/Ti;  %before...[x,y]=meshgrid(iN*Tf,iN*Tf);
        fase_f=exp(1i*k*d)/(1i*lamb*d)*exp(1i*k/2/d*(x.^2+y.^2));
        Uf=fase_f.*integ; %electric field values at position d
    end
```