Exact ray tracing in MATLAB

Maria Ruiz-Gonzalez

Introduction

This tutorial explains how to program a simple geometric ray tracing program in MATLAB, which can be written in any other programming language like C or Python and extended to add elements and complexity. The main purpose is that the student understands what a ray tracing software like Zemax or Code V does, and that the analysis can be performed even if there's no access to any of those software.

The code consists of a main program and a function for a plano-convex lens that in turn consists of another two functions, one for the refraction at a spherical surface and a second one for refraction at a plane surface, as explained in figure 1. Building the program in functions makes easier to debug and understand since the code is simpler, and since each surface is a different function, different elements can be easily constructed.

By using a for-loop, it is possible to compute many different rays at the same time, which are stored in the rows of a matrix. That matrix has all the information necessary to perform different analyses. The inputs for the plano-convex lens function are the height of the ray, refractive index, thickness of lens, radius of spherical surface, and resolution for computations in the z-axis. The output is a ray matrix and some vectors related to the z-axis that make easier the visualization of the results.

Finally, two examples of what can be analyzed with the ray matrix are given, that consists of a few extra lines of code. It is important to notice that even if we choose to do paraxial optics for simplicity for a first-order approximation of an optical system, an exact ray trace is also simple and gives us substantially more information, for instance, spherical aberration and ray fan plots.

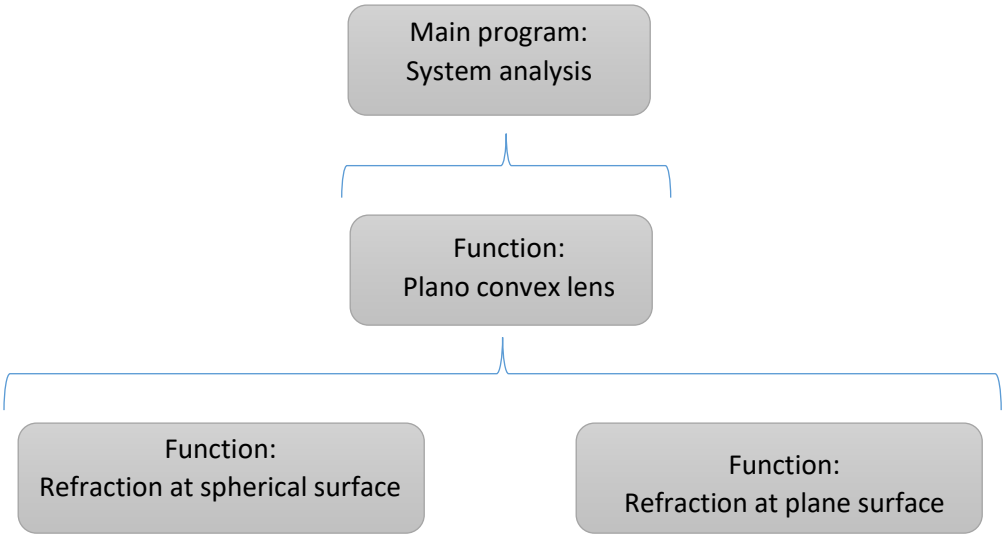

Figure 1. Structure of the geometric ray tracing program for a plano-convex lens.

Refraction at plane interface

The refraction at an interface is described by the Snell's law:

$$n \sin\theta = n' \sin\theta'$$

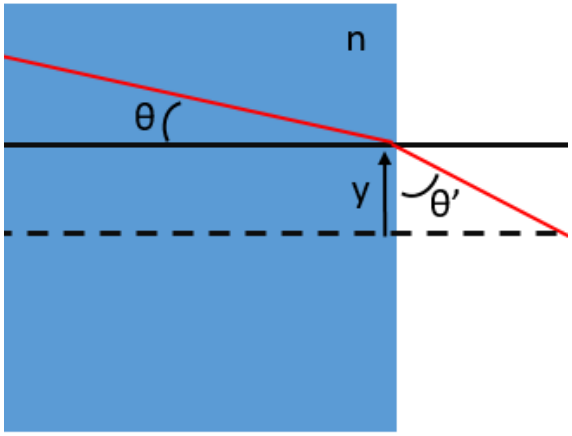

Figure 2. Refraction at a plane interface described by Snell's law.

The Matlab function for refraction at a plane interface takes as input height *y* of the ray at the interface, slope $u = \tan\theta$, thickness of the lens, index of refraction *n*, and vector *z*, which is used to plot the ray in air (back of lens). In our case, the ray travels from the medium to air.

```
function [ray_air] = plane_refract_ray(y,slope,thickness,n,z)

    theta1 = atan(slope);
    theta2 = asin(n*sin(theta1));
    slope2 = tan(theta2);
    ray_air = (z-thickness)*slope2 + y;

end
```

Refraction at spherical interface

We need to compute the slope $u = \tan\theta$ inside the lens, using the geometry in figure 3.

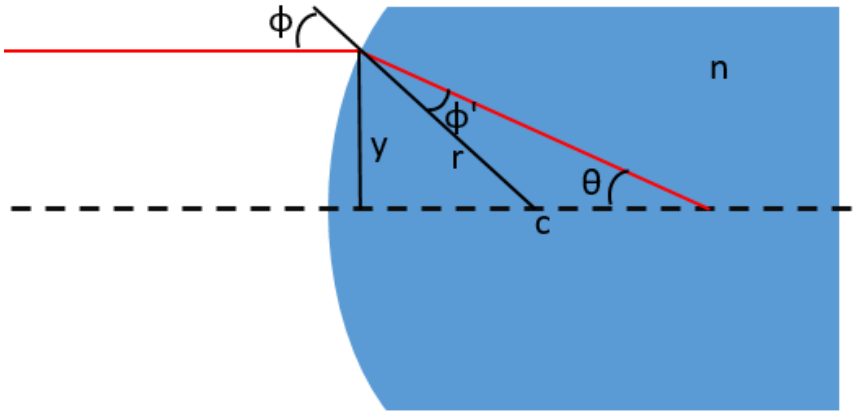

Figure 3. Refraction of incoming ray at spherical interface.

It is easy to see that:

$$\sin\phi = \frac{y}{r}$$

By Snell's law:

$$n\sin\phi' = \sin\phi \rightarrow \sin\phi' = \frac{\sin\phi}{n}$$

Finally, after some simple geometry:

$$\theta = \phi' - \phi$$

For the Matlab function, input variables are height *y*, radius *r*, thickness of the lens, index of refraction *n* and step size *Δz*. Outputs are the ray inside the lens, slope and *z* axis. In this case, the ray travels from air to the medium.

```
function [ray,slope,z] = sphere_refract_ray(y,radius,thickness,n,dz)

    sag = radius - sqrt(radius^2 - y^2); %lens sag at y
    z = sag:dz:thickness;
    sin_phi1 = y/radius;
    sin_phi2 =  sin_phi1/n;
    phi1 = asin(sin_phi1);
    phi2 = asin(sin_phi2);
    theta = phi2-phi1;
    slope = tan(theta);
    ray = slope*(z-sag) + y; % Ray in lens

end
```

Plano-convex lens

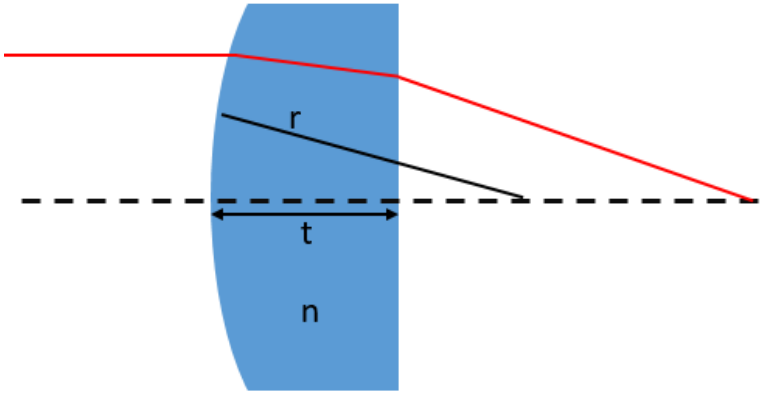

Figure 4. Geometry of the plano-convex lens solved in this tutorial.

In order to construct the rays through the lens, we have to use the two functions described above, in the correct order. The paraxial focal length is computed for visualization purposes. The function for the plano-convex lens takes as input the index of refraction of the lens, radius of first surface, thickness of lens, step size Δz and height y. The results for all rays are stored in a matrix, where each row is one ray.

```
function [raymatrix,z_front,z_optaxis,zmax] = plano_convex(n,radius,thickness,dz,y)

    power = (n-1)/radius; %lens power
    f = 1/power; %paraxial focal length

    zmax = floor(f+.1*f); %end of z-axis
    z_front = 0:dz:thickness-dz; %z-axis back of plane surface
    z_back = thickness:dz:zmax-dz; %z-axis front of plane surface
    z_optaxis = [z_front,z_back]; %total optical axis

    y(y==0)=10^-10;
    raymatrix = zeros(length(y),length(z_optaxis));

    %Ray tracing
    for i = 1:length(y)

        %Refraction at spherical surface
        [ray_lens, slope, x_lens] = sphere_refract_ray(y(i),radius,thickness,n,dz);

        %Refraction at plane surface
        [ray_air] = plane_refract_ray(ray_lens(end),slope,thickness,n,z_back);

        %Incoming ray
        x_front_air = 0:dz:x_lens(1)-dz;
        ray_front_air = y(i)*ones(1,length(x_front_air));

        %Create matrix of rays (adjust length if necessary)
        if length(ray_lens)+length(ray_air)+length(x_front_air) <= length(z_optaxis)
            raymatrix(i,:) = [ray_front_air,ray_lens,ray_air];
        else
            raymatrix(i,:) = [ray_front_air, ray_lens(1:length(ray_lens)-1), ray_air];
        end

    end
end
```

Main program

The main program is where all the analysis is performed using the information of the height of each ray at all points obtained with the plano-convex lens function. It's here where the programmer can decide what to visualize and the analyses to perform. The example shown here is for a plano-convex lens of radius R = 20 mm, center thickness = 2 mm and index of refraction n = 1.5168. The next few lines of code are the initialization variables and the visualization of the system.

```
n = 1.5168; %Index of refraction of lens
radius = 20; %Radius of spherical surface
thickness = 2; %Central thickness of lens
dz = 0.01; %Step size for computation purposes
aperture = 5;
number_rays = 11;
dy = (2*aperture + 1)/number_rays;
y = -aperture:dy:aperture; %Field of view

%Ray matrix
[raymatrix,x_front,x_optaxis,zmax] = plano_convex(n,radius,thickness,dz,y);

%Figure
front_lens = sqrt(radius^2 - (x_front-radius).^2);
figure(1)
plot(x_optaxis,raymatrix','r') %Rays
hold on
% Lens back surface
line([thickness thickness], [max(front_lens) -max(front_lens)],'color','b')
plot(x_front,front_lens,'b',x_front,-front_lens,'b') %Lens front surface
plot(x_optaxis,zeros(1,length(x_optaxis)),'k--') %Optical axis
hold off
axis([-thickness zmax -max(front_lens) max(front_lens)])
```

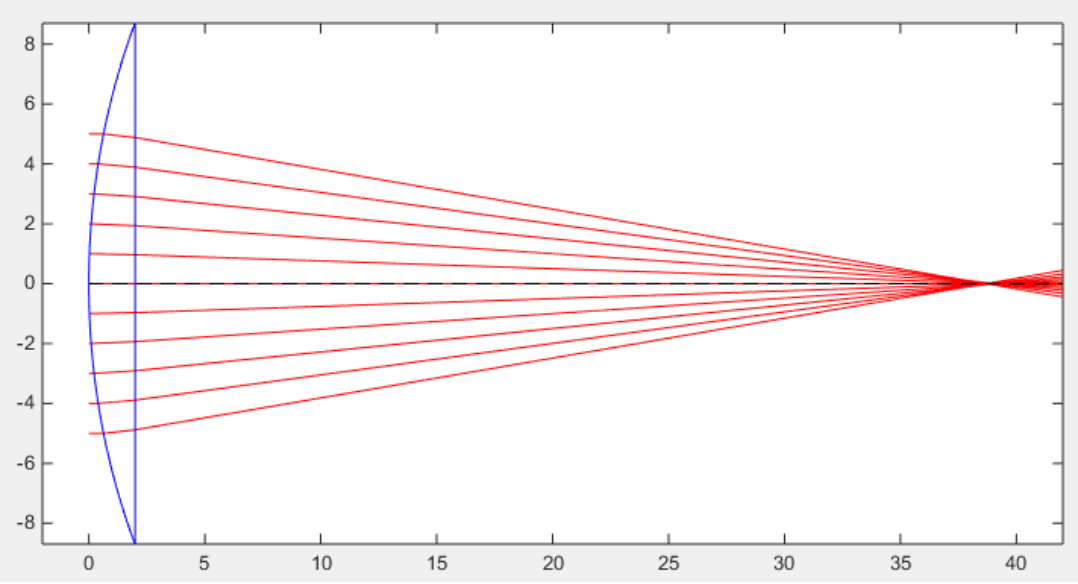

Figure 5. Lens computed with the plano-convex lens function and ray traces for collimated incoming rays.

Example: Ray fan plot

Each column of the ray matrix corresponds to the heights *y* of all rays at a constant *z*. In order to compute a ray fan plot, we need to find where the paraxial ray crosses the optical axis, and plot the column corresponding to that point.

```
%Find where each ray crosses optical axis
ray_focus = zeros(1,length(y));
for i = 1:length(y)
    [a, ray_focus(i)] = min(abs(raymatrix(i,:)));
End

%Find where paraxial ray crosses optical axis
paraxial_focus = ray_focus(ceil(length(y)/2));

%Ray fan plot
plot(y,raymatrix(:,paraxial_focus)')
```

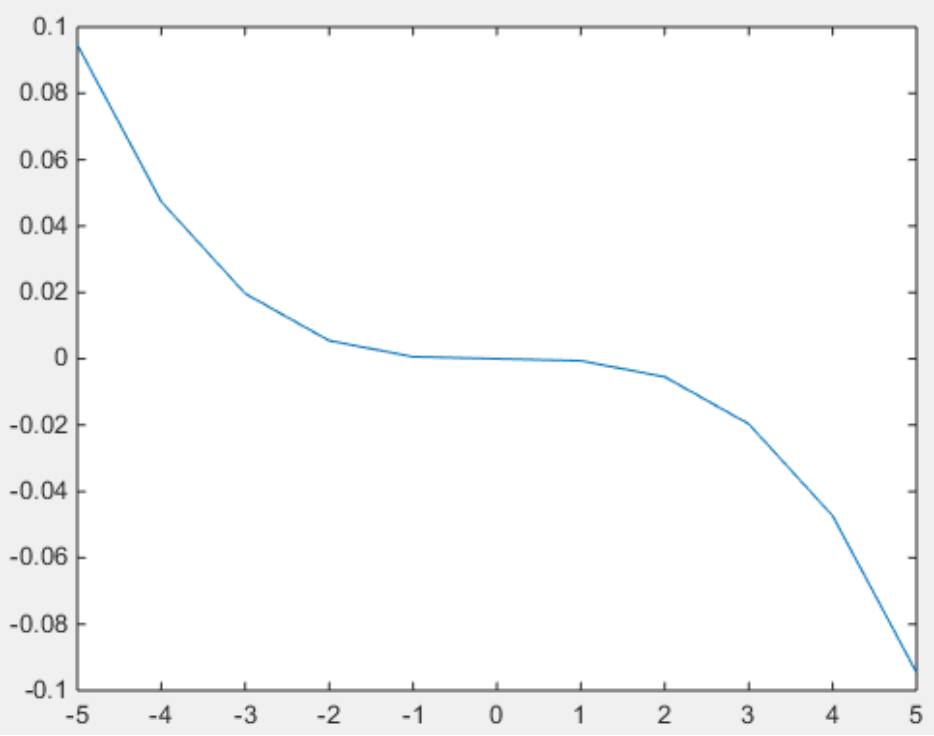

Figure 6. Ray fan plot for the plano-convex lens.

Example: Longitudinal spherical aberration

In the previous example, it was found where the rays cross the optical axis. We can plot those values to see the longitudinal spherical aberration of the system.

```
spher_ab = x_optaxis(ray_focus(find(y>=0)));
plot(spher_ab - spher_ab(1),y(find(y>=0)))
```

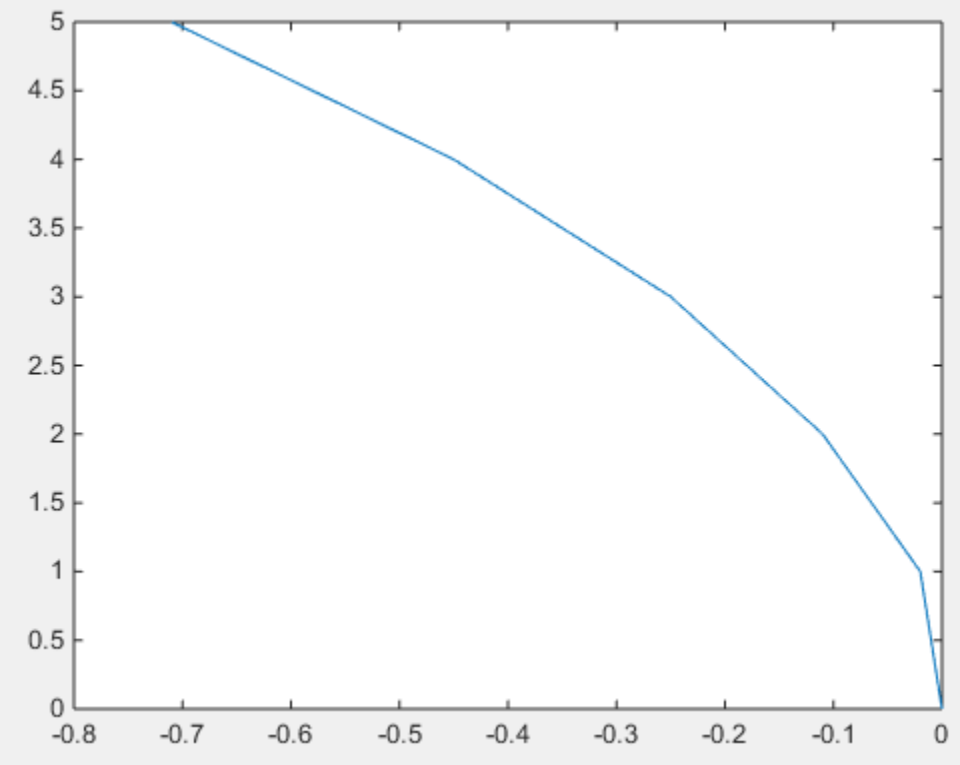

Figure 7. Longitudinal spherical aberration of the plano-convex lens.

Appendix A

Code to obtain the results presented in this tutorial. Save each function in different files, the name of the file must be the name of the function, e.g. `sphere_refract_ray.m`.

```
function [ray,slope,z] = sphere_refract_ray(y,radius,thickness,n,dz)
    sag = radius - sqrt(radius^2 - y^2); %lens sag at y
    z = sag:dz:thickness;
    sin_phi1 = y/radius;
    sin_phi2 =  sin_phi1/n;
    phi1 = asin(sin_phi1);
    phi2 = asin(sin_phi2);
    theta = phi2-phi1;
    slope = tan(theta);
    ray = slope*(z-sag) + y; % Ray in lens
end
```

```
function [ray_air] = plane_refract_ray(y,slope,thickness,n,z)

    theta1 = atan(slope);
    theta2 = asin(n*sin(theta1));
    slope2 = tan(theta2);
    ray_air = (z-thickness)*slope2 + y;

end
```

```
function [raymatrix,z_front,z_optaxis,zmax] = plano_convex(n,radius,thickness,dz,y)
    power = (n-1)/radius; %lens power
    f = 1/power; %paraxial focal lenght
    zmax = floor(f+.1*f);
    z_front = 0:dz:thickness-dz; %x-axis back of plane surface
    z_back = thickness:dz:zmax-dz; %x-axis front of plane surface
    z_optaxis = [z_front,z_back]; %total optical axis
    y(y==0)=10^-10;
    raymatrix = zeros(length(y),length(z_optaxis));

    %Ray tracing
    for i = 1:length(y)

        %Refraction at spherical surface
        [ray_lens, slope, x_lens] = sphere_refract_ray(y(i),radius,thickness,n,dz);

        %Refraction at plane surface
        [ray_air] = plane_refract_ray(ray_lens(end),slope,thickness,n,z_back);

        %Ray comming in
        x_front_air = 0:dz:x_lens(1)-dz;
        ray_front_air = y(i)*ones(1,length(x_front_air));

        %Create matrix of rays (adjust length if necessary)
        if length(ray_lens)+length(ray_air)+length(x_front_air) <= length(z_optaxis)
            raymatrix(i,:) = [ray_front_air,ray_lens,ray_air];
        else
            raymatrix(i,:) = [ray_front_air, ray_lens(1:length(ray_lens)-1), ray_air];
        end

    end
end
```

```
%% Plano-convex lens: ray tracing analysis

clear all

n = 1.5168; %Index of refraction of lens
radius = 20; %Radius of spherical surface
thickness = 2; %Central thickness of lens
dz = 0.01; %Step size for computation purposes
%dy = 1; %Separation between rays
aperture = 5;
number_rays = 11;
dy = (2*aperture + 1)/number_rays;
y = -aperture:dy:aperture; %Field of view


%Ray matrix
[raymatrix,x_front,x_optaxis,zmax] = plano_convex(n,radius,thickness,dz,y);

%Figures
front_lens = sqrt(radius^2 - (x_front-radius).^2);
figure(1)
plot(x_optaxis,raymatrix','r') %Rays
hold on
line([thickness thickness], [max(front_lens) -max(front_lens)],'color','b')
%Lens back surface
plot(x_front,front_lens,'b',x_front,-front_lens,'b') %Lens front surface
plot(x_optaxis,zeros(1,length(x_optaxis)),'k--') %Optical axis
hold off
axis([-thickness zmax -max(front_lens) max(front_lens)])


%% Ray fan plots
%Find where each ray focuses
ray_focus = zeros(1,length(y));
for i = 1:length(y)
    [a, ray_focus(i)] = min(abs(raymatrix(i,:)));
end
%Find where paraxial ray focuses
paraxial_focus = ray_focus(ceil(length(y)/2));
figure(2)
%Ray-fan plot
plot(y,raymatrix(:,paraxial_focus)')


%% Spherical aberration
figure(3)
spher_ab = x_optaxis(ray_focus(find(y>=0)));
plot(spher_ab - spher_ab(1),y(find(y>=0)))
```