

Introduction to Geometrical Optics - a 2D ray tracing Excel model for spherical mirrors - Part 3

by George Lungu

- While the previous section dealt with defining the parameters and geometry of the input beam and solving the system of equation to find the formulas of the coordinates of the points where the incident rays meet the mirror, this section creates a VBA custom function to calculate those coordinates.
- This is an exact model in the sense that no geometrical approximations are used, however the model does not take into consideration diffraction effects.

Reflect() - new VBA function:

- In the previous section we solved the quadratic system of equations leading to finding the exact Cartesian coordinates of the incidence points on the mirror surface.
- Now we will write a custom VBA function to implement those results and call it "Reflect".
- This custom spreadsheet function will return four numbers: two x-y Cartesian coordinates of the point in space where the ray hits the mirror surface, the tangents of the incident angle and emergent angle.
- The next version of this function must be easy to be cascaded when later used for successive reflection and refraction models

[<www.excelunusual.com>](http://www.excelunusual.com)



First 8-m mirror blank for the Gemini project. The mirror blanks are manufactured by fusing hexagonal pieces of ULE zero-expansion glass in a high-temperature furnace.

Review of the formulas behind the "Reflect()" custom function:

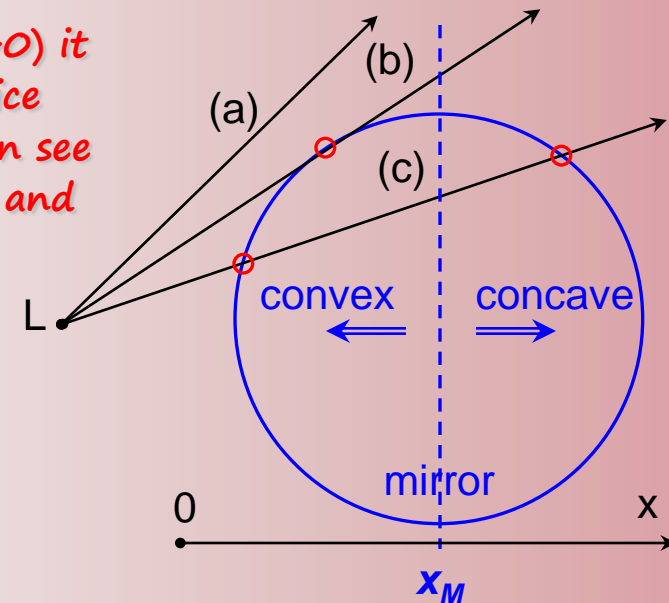
The constants a , b , c used in the formulas to the left of the page are defined below:

$$\begin{cases} y_{1,2} = \tan(\alpha_i) \cdot x_{1,2} + y_L - x_L \cdot \tan(\alpha_i) \\ x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4 \cdot a \cdot c}}{2 \cdot a} \end{cases} \begin{cases} a = \frac{1}{\cos^2(\alpha_i)} \\ b = 2 \cdot [\tan(\alpha_i) \cdot (y_L - y_M - x_L \cdot \tan(\alpha_i)) - x_M - R] \\ c = (x_M + R)^2 + (y_L - y_M - x_L \cdot \tan(\alpha_i))^2 - R^2 \end{cases}$$

-Based on the previous presentation if the mirror is convex ($R > 0$) it means we choose the smaller x (the solution with minus) and vice versa. This is obvious also from the diagram to the right. We can see that the smaller x corresponds to light hitting a convex mirror and a larger x corresponds to light hitting a concave mirror.

- In this case the formulas become:

$$\begin{cases} x = \frac{-b - \text{sign}(R) \cdot \sqrt{b^2 - 4 \cdot a \cdot c}}{2 \cdot a} \\ y = \tan(\alpha_i) \cdot x + y_L - x_L \cdot \tan(\alpha_i) \end{cases}$$



Even if there might be solutions determined by the ray intersecting the mirror circle, maybe the diameter of the mirror "d" will prevent that to happen and the ray will pass undeviated. We will deal with this situation later and for now let's implement a preliminary custom VBA function.

Implementation of a preliminary "Reflect_1()" custom VBA function:

- Let's implement a preliminary function which has as arguments the Cartesian coordinates of the light source L (x_L , y_L), the coordinates of the mirror vertex M (x_M , y_M), the angle of the incident ray with respect to the horizontal "alpha_incident" and the mirror radius "R".
- For now, this function will only return the Cartesian coordinates of the point of incidence I (x_I , y_I).

- Rename the current worksheet "Tutorial_1+2"

- Copy the current worksheet and rename the copy "Tutorial_3"

- In the VBA editor insert a module and in that module write the following code:

- These are just the formulas from the previous page set up as a custom VBA function.

-The output of this function is a 2D vector array and we will need to be careful when we type it in (using F2 + Ctrl + Shift + Enter).

- The correctness of the function will be verified by plotting a series of incident point coordinates on the same chart with the mirror.

```
Function Reflect_1(xL, yL, xM, yM, alpha_incident, R)
```

```
    Dim a, b, c As Double
```

```
    a = 1 / Cos(alpha_incident) ^ 2
```

```
    b = 2 * (Tan(alpha_incident) * (yL - yM - xL * Tan(alpha_incident)) - xM - R)
```

```
    c = (xM + R) ^ 2 + (yL - yM - xL * Tan(alpha_incident)) ^ 2 - R ^ 2
```

```
    x = (-b - Sgn(R) * Sqr(b ^ 2 - 4 * a * c)) / (2 * a)
```

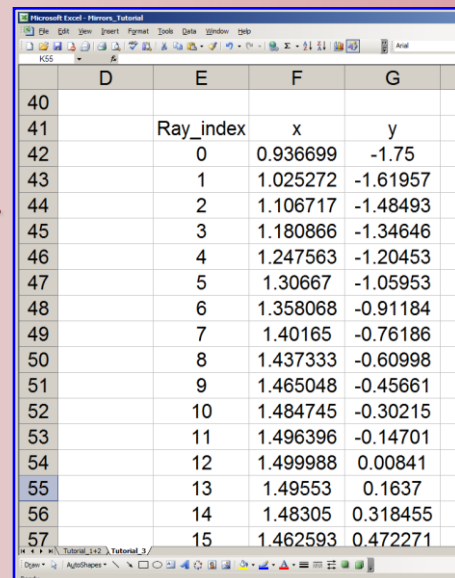
```
    y = Tan(alpha_incident) * x + yL - xL * Tan(alpha_incident)
```

```
    Reflect_1 = Array(x, y)
```

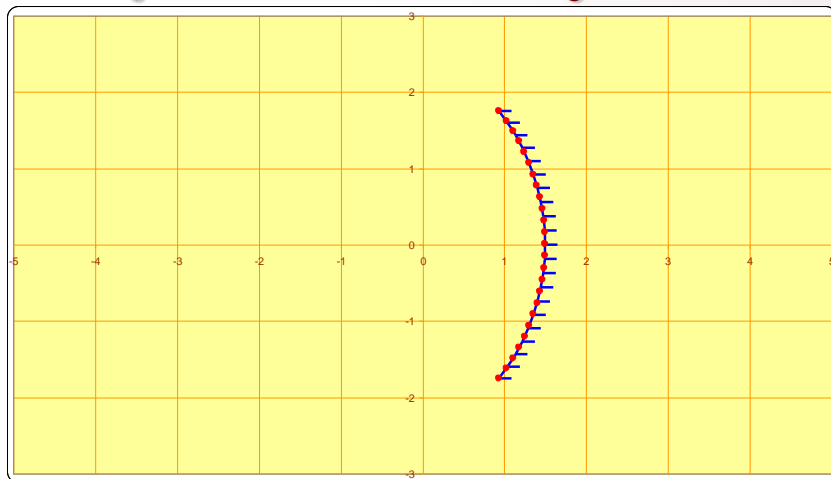
```
End Function
```


Verifying the functionality of the Reflect 1() custom function:

- In the worksheet "Tutorial_3" in the range E42:E66 we will create a "Ray_Index" column with integer numbers running from 0 to 24. I have chosen a 25 ray input beam since 25 is different from 21 which is the number of hachure lines on the back of the mirror (I didn't want people to inadvertently associate a ray to each hachure dash).
- Cell E42: "=", cell E43: "=", then copy E43 down to cell E66.
- Cell F24: "Reflect_1(xL,yL,xM,yM,alpha_min+E42*delta_alpha,Radius)
- Select range F42:G42 and then holding F2 down press Ctrl+Shift+Enter and you will have the 2D output vector of the custom function in range F42:G42.
- Copy range F42:G42 down to range F66:G66
- Add a series to the chart having the x-data taken from range F42:F66 and the y-data taken from range G421:G66.



	D	E	F	G
40				
41		Ray_index	x	y
42		0	0.936699	-1.75
43		1	1.025272	-1.61957
44		2	1.106717	-1.48493
45		3	1.180866	-1.34646
46		4	1.247563	-1.20453
47		5	1.30667	-1.05953
48		6	1.358068	-0.91184
49		7	1.40165	-0.76186
50		8	1.437333	-0.60998
51		9	1.465048	-0.45661
52		10	1.484745	-0.30215
53		11	1.496396	-0.14701
54		12	1.499988	0.00841
55		13	1.49553	0.1637
56		14	1.48305	0.318455
57		15	1.462593	0.472271



- We can see that the new data (red points in the snapshot) overlap with the mirror profile and this is what we intended.

- While deriving the formula I made a small mistake forgetting the R^2 in the formula of "c". It took me 2 hours to figure it out and I recommend that you don't give up if things don't work for the first time. Persist and you will be rewarded!

Deriving the emergent ray angle formulas:

- Emergent rays are the rays leaving the mirror after reflection.
- We need these formulas to write the equations of the emerging rays so that we can plot them.
- We named the center of curvature of the mirror "C" and the incidence point "P".
- Since C is the center of curvature, the distance CP is equal to the radius of the mirror.
- We also know that the radius of a circle is normal to its circumference.

From the reflection laws we have: $\beta_I = \beta_E$

From the diagram we can write the following angle equality:

$$\beta_I = \widehat{CPP'} - \widehat{LPP''}$$

and since triangles CPP' and LPP'' are straight triangles we can write the following relationships:

$$\widehat{CPP'} = \frac{\pi}{2} - \alpha_p \quad \text{and} \quad \widehat{LPP''} = \frac{\pi}{2} - \alpha_i$$

from the above

expressions we can therefore write:

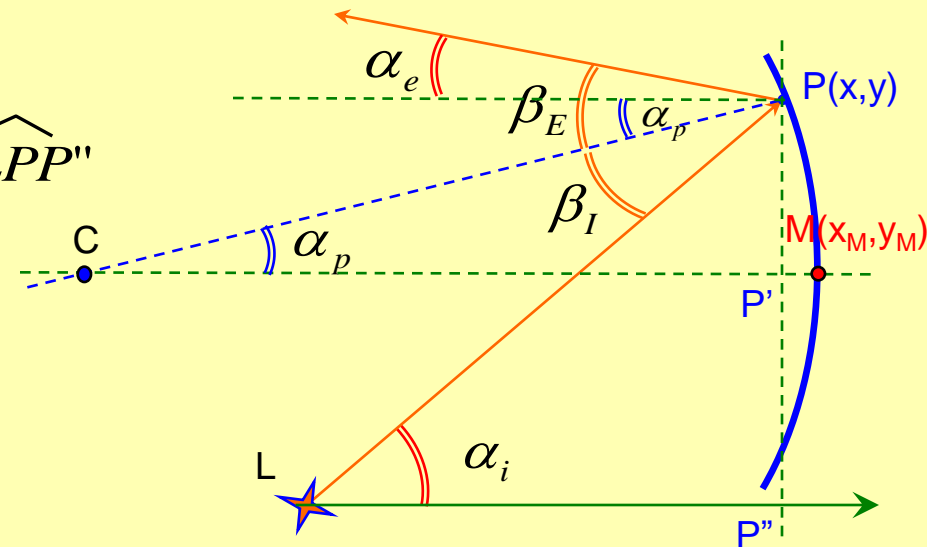
$$\beta_I = \beta_E = \alpha_i - \alpha_p$$

We get the new emergent ray angle:

$$\alpha_e = \alpha_i - 2 \cdot \alpha_p$$

And expressing the emergent ray angle as:

$$\alpha_e = \beta_E - \alpha_p$$



We know α_i for every ray and we can calculate α_p from the coordinate of point P (which we know too). There is a sign change in front of the number "2" and it comes from the negative R value in the depicted case:

$$\alpha_e = \alpha_i + 2 \cdot \text{asin}\left(\frac{y_P - y_M}{R}\right)$$

- Looking at the formula in the previous page we realize that the sign is correct since for a positive angle the emerging ray climbs.
- Let's see how to add this formula to the new "Reflect()" custom VBA function.

$$\alpha_e = \alpha_i + 2 \cdot \text{asin}\left(\frac{y_P - y_M}{R}\right)$$

Upgrading the "Reflect 1()" custom VBA function to "Reflect()":

```
Function Reflect(xL, yL, xM, yM, alpha_incident, R)
    Dim a, b, c As Double
    a = 1 / Cos(alpha_incident) ^ 2
    b = 2 * (Tan(alpha_incident) * (yL - yM - xL * Tan(alpha_incident)) - xM - R)
    c = (xM + R) ^ 2 + (yL - yM - xL * Tan(alpha_incident)) ^ 2 - R ^ 2
    x = (-b - Sgn(R) * Sqr(b ^ 2 - 4 * a * c)) / (2 * a)
    y = Tan(alpha_incident) * x + yL - xL * Tan(alpha_incident)
    z = alpha_incident + 2 * Application.Asin((y - yM) / R)
    Reflect = Array(x, y, z)
End Function
```

- The code to the left creates the new custom function which will return not only the x-y coordinates of the incidence point but also the slope of the emergent (reflected ray).

to be continued...