Project 1:

Reconstructing Phase Function and Reflecting Surface with Intensity of Light Rays

Jenny Be and Brian Vazquez

The Objective

Want to construct a system mathematically to understand how a reflected light wave from a surface is used to find the shape of the surface and then study the connections between the shaping of wavefronts and the design of creating such system.

The Problem

The general harmonic wave equation has the form

$$\Psi(\vec{r},t) = a(\vec{r})\cos[\omega t - k_0\phi(\vec{r})]$$

where $\vec{r}$ is the position vector (x,y,z), $a(\vec{r})$ is the amplitude of the wave, ω is the angular temporal frequency, k_0 is the wavenumber in a vacuum, and $\phi(\vec{r})$ is the phase function. The intensity of light rays, I, is equal to the amplitude of the light ray squared, $[a(\vec{r})]^2$. From I data, $-I_z = f(x,y)$ can be found. With the I and $f(x,y)$ known, the Transport of Intensity Equation (TIE) with Neumann boundary condition (equation below) can be solved to find the $P(x,y)$ values.

$$\begin{cases} \nabla\cdot(I\nabla P) = -I_z = f \ in \ D \\ I\frac{\partial P}{\partial n} = 0 \ on \ \partial D \end{cases}$$

After solving the TIE problem, the $P(x,y)$ values found is used to reconstruct the phase function since $\phi(\vec{r}) = z + P(x,y)$. To get a wavefront from this, we set $\phi(\vec{r}) = k$ where k is a constant. Reflected wavefronts are useful for reconstructing an unknown surface since it will have a similar shape to the surface, but will not look exactly the same.

The gradient of the phase function, $\nabla\phi = [P_x, P_y, 1]$, gives the direction of the light rays. The direction of light rays cross the wavefronts orthogonally. By the Eikonal's Equation, $|\nabla\phi| = n$ the unit vector of the light ray direction is

$$\vec{v} = \frac{\nabla\phi}{|\nabla\phi|} = \frac{\nabla\phi}{n}$$

where n is the refraction index. For simplicity of calculations, assume the medium used in this experiment is air and that $n_{air} = 1$.

Reconstructing Phase Function and Wavefronts

This method will be a simplistic model on how to reconstruct a reflected phase function based off of the intensity of the reflected light rays. When the light ray is reflected off the surface, it is assumed that the intensity of the rays can be collected. With the collected intensity, the height of the surface can be approximated which can be used to reconstruct the phase function.

To figure out the f for the partial differential equation, we need to find I_z by collecting the intensity of the reflected light rays at multiple heights. With the different intensity instances collect, use finite differences to compute I_z. Since $-I_z = f$, we can use TIENeumann to calculate the $P(x, y)$ value with $z = sensor_z$ as the height of the sensor to reconstruct phase function given $\phi(x, y, z) = z + P(x, y)$.

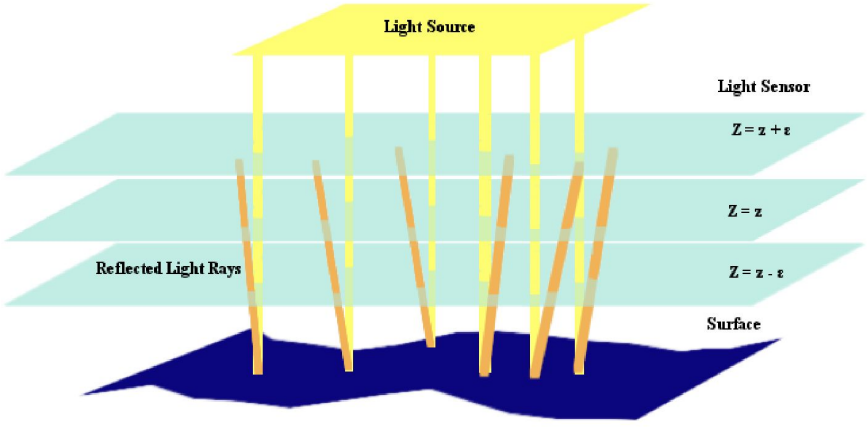


Illustration of how to collect data to calculate I_z

[num] = index(m, i, j)

The index function is used to map each node index of an $N \times M$ mesh to a numerical value based on the number of columns, m, and the index of the node (i, j). So, the indices

$$(1,1) \to (1,2) \to \cdots \to (1,M) \to (2,1) \to \cdots \to (N, M-1) \to (N,M)$$

becomes

$$1 \to 2 \to \cdots \to M \to M+1 \to \cdots \to N \times M - 1 \to N \times M$$

[p, lambda] = TIENeumann(I, f, h, sigma)

The TIENeumann function is used to solve the TIE problem on a rectangular domain. The input values I and f are $N \times M$ matrices that represents the intensity and the values of $f(x, y)$ on the domain $\bar{D}$. The input h is the number that represents the mesh size for both the x and y direction.

Sigma is a number that represents the sum of $P(x,y)$ over $\bar{D}$. The function first builds the $\bar{A}$ and $\bar{F}$ matrices and then solves $\bar{A}\bar{P} = \bar{F}$ where $\bar{A}$, $\bar{P}$, and $\bar{F}$ is defined as

$$\bar{A} = \begin{bmatrix} A & 1 \\ 1^T & 0 \end{bmatrix} \qquad \bar{P} = \begin{bmatrix} \vec{P} \\ \lambda \end{bmatrix} \qquad \bar{F} = \begin{bmatrix} \vec{F} \\ \sigma \end{bmatrix}$$

When building the $\bar{A}$ and $\bar{F}$, the function needs the index function to convert the matrix index into the corresponding numerical value.

The $\bar{A}$ is an $(NM+1)\times(NM+1)$ matrix with $A_{MN\times MN}$ matrix and $N\times 1$ vector of ones. When building the $\bar{A}$ matrix, there are five different schemes to consider for the boundary and inner nodes using the data from the intensity collected. Let $I_{i,j} = I(x_i, y_j)$, $P_{i,j} = P(x_i, y_j)$, and $f_{i,j} = f(x_i, y_j)$, then the scheme can be represented as

Right Boundary: $$-I_{i,m}P_{i,m-2} + 4I_{i,m}P_{i,m-1} - 3I_{i,m}P_{i,m} = 0$$

Left Boundary: $$-I_{i,1}P_{i,1} + 4I_{i,1}P_{i,2} - 3I_{i,1}P_{i,3} = 0$$

Bottom Boundary: $$-3I_{1,j}P_{1,j} + 4I_{1,j}P - 2, j - I_{1,j}P_{3,j} = 0$$

Top Boundary: $$-3I_{n,j}P_{n,j} + 4I_{n,j}P_{n-1,j} - I_{n,j}P_{n-2,j} = 0$$

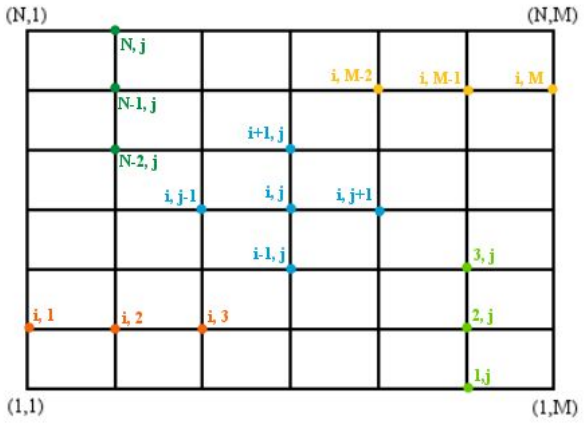
Interior: $$A_1P_{i-1,j} + A_2P_{i,j-1} - A_3P_{i,j} + A_4P_{i,j+1} + A_5P_{i+1,j} = 2h^2 f_{i,j}$$

Where

$$A_1 = I_{i,j} + I_{i-1,j} \qquad A_2 = I_{i,j} + I_{i,j-1}$$
$$A_3 = I_{i,j+1} + I_{i,j-1} + I_{i+1,j} + I_{i-1,j} + 4I_{i,j}$$
$$A_4 = I_{i,j+1} + I_{i,j} \qquad A_5 = I_{i+1,j} + I_{i,j}$$

$\bar{F}$ is a $(NM+1)\times 1$ vector using the f matrix in a vector form. The indices that corresponds to the boundary of the domains will have a 0 value while the indices that corresponds to the interior of the domain will have the value $2h^2 f_{i,j}$.

Once the $\bar{A}$ and $\bar{F}$ is constructed, the function will solve for $\bar{P}$. The last value of $\bar{P}$ gives the value of lambda. Since the $\vec{P}$ of $\bar{P}$ is a vector, the function then converts the $\vec{P}$ to a matrix form, P. The function then outputs P and lambda. The resulting P matrix is a $N\times M$ matrix that stores the $p(x,y)$ values on $\bar{D}$ while lambda is the number that measures the validity of the compatibility condition.



Stencil of the boundary and interior schemes to build matrix A

Accuracy of the Method

The method used to calculate our function $P(x, y)$ from the partial differential equation using intensity data is only good for theoretical purposes. In order to acquire a practical use of it, we need to discretize the PDE so that a system can be built to gather Intensity data and form a numerical method as done in previous section. We first get the left side of our PDE and change it to the form:

$$(IP_x)_x + (IP_y)_y = f$$

Where our left component is $A = (IP_x)_x$ and our right component is $B = (IP_y)_y$. The numerical expansion for component A is:

$$\frac{(I_{i,j+1} + I_{i,j})(P_{i,j+1} - P_{i,j}) - (I_{i,j} + I_{i,j-1})(P_{i,j} - P_{i,j-1})}{2h^2}$$

We now perform Taylor expansion on $I_{i\pm1,j}, I_{i,j\pm1}$ and $P_{i\pm1,j}, P_{i,j\pm1}$ up to $O(h^5)$:

$$I_{i,j+1} = I + hI_y + \frac{h^2}{2!}I_{yy} + \frac{h^3}{3!}I_{yyy} + \frac{h^4}{4!}I_{yyyy} + O(h^5)$$

$$I_{i,j-1} = I - hI_y + \frac{h^2}{2!}I_{yy} - \frac{h^3}{3!}I_{yyy} + \frac{h^4}{4!}I_{yyyy} + O(h^5)$$

$$I_{i+1,j} = I + hI_x + \frac{h^2}{2!}I_{xx} + \frac{h^3}{3!}I_{xxx} + \frac{h^4}{4!}I_{xxxx} + O(h^5)$$

$$I_{i-1,j} = I - hI_x + \frac{h^2}{2!}I_{xx} - \frac{h^3}{3!}I_{xxx} + \frac{h^4}{4!}I_{xxxx} + O(h^5)$$

$$P_{i,j+1} = P + hP_y + \frac{h^2}{2!}P_{yy} + \frac{h^3}{3!}P_{yyy} + \frac{h^4}{4!}P_{yyyy} + O(h^5)$$

$$P_{i,j-1} = P - hP_y + \frac{h^2}{2!}P_{yy} - \frac{h^3}{3!}P_{yyy} + \frac{h^4}{4!}P_{yyyy} + O(h^5)$$

$$P_{i+1,j} = P + hP_x + \frac{h^2}{2!}P_{xx} + \frac{h^3}{3!}P_{xxx} + \frac{h^4}{4!}P_{xxxx} + O(h^5)$$

$$P_{i-1,j} = P - hP_x + \frac{h^2}{2!}P_{xx} - \frac{h^3}{3!}P_{xxx} + \frac{h^4}{4!}P_{xxxx} + O(h^5)$$

where $I = I_{i,j}$ and $P = P_{i,j}$.

We then put back into A and combine like terms with respect to h and note that the $O(h)$ and $O(h^3)$ cancel out and left with:

$$A = (IP_{yy} + P_y I_y) + O(h^2)$$

We note that $(IP_{yy} + P_y I_y) = (IP_y)_y$.

Thus, having component A of $O(h^2)$. For component B, we have:

$$\frac{(I_{i+1,j} + I_{i,j})(P_{i+1,j} - P_{i,j}) - (I_{i,j} + I_{i-1,j})(P_{i,j} - P_{i-1,j})}{2h^2}$$

We note that this component has the same form as A except that we have the $I_{i\pm1,j}$ and $P_{i\pm1,j}$ parts. This gives us a result of:

$$B = (IP_{xx} + P_x I_x) + O(h^2)$$

And also note that

$$(IP_{xx} + P_x I_x) = (IP_x)_x$$

Thus our scheme has the form:

$$(IP_x)_x + (IP_y)_y + O(h^2) = f$$

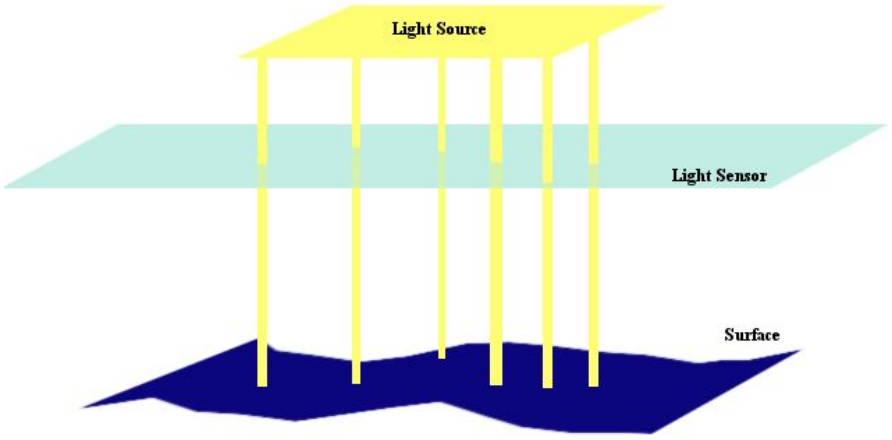
And hence of second order accuracy.

Reconstructing Reflective Surface

There are many methods to reconstruct the surface from reflected light rays. We will use a system where we shoot rays perpendicular to a rectangular plane D fixed at $z = \lambda$ that is parallel to the (x, y)-plane. Our light sensor is placed in between our light source D and our surface $z = f(x, y)$ at $z = \alpha$. Here, we assume that we are close enough to our surface such that, for some small number β, the surface we wish to map oscillates:

$$-\beta \leq z = f(x, y) \leq \beta$$

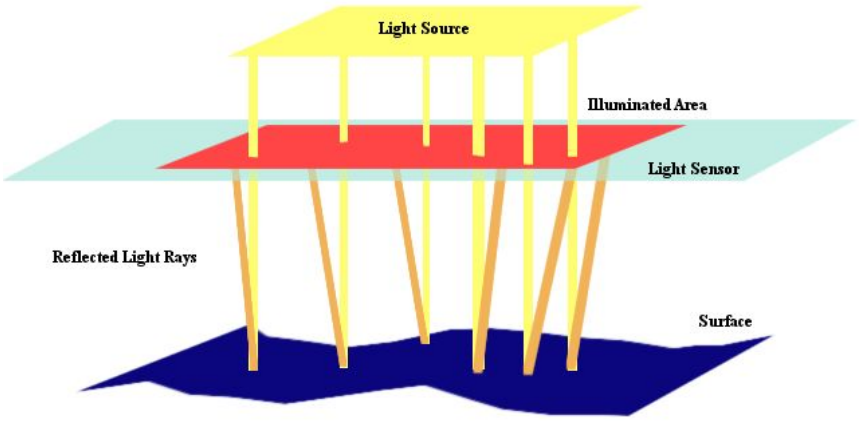
This ensure that we map flatter surface so that we don't scatter light too much so that the light sensor is able to capture incident rays consistently with their source rays. This ensures that our TIENeuman function is able to maintain its $O(h^2)$ accuracy.



Basic setup of the experiment

First, we assume that the light rays reflecting off of the surface are not immediately crossing each other so that points captured on the wavefront remain consistent with each other in order of their incidents. Thus, on the wavefront, the first point would belong to the first incident off the surface and the second point on the wavefront would belong to the second incident, and so on.

To find the shape of the reflecting surface, we first look at the behavior of the wavefront with the light rays passing through them. Since we know that the reflected light ray crosses a wavefront at some point, $\phi(\vec{r}) = k_0$, with direction $\vec{v} = \nabla\phi$, we can find the point where the vector $-\vec{v}$ crosses the boundary $(x, y) \in [a, b] \times [c, d], z = z_0$, as shown below:



When the reflected light hits the senor, a illuminated area is formed

Our main goal here is to find the point $P_0 = (x_0, y_0, z_0)$ where the only unknown is z_0. We do this for every point along our rectangular domain $[a, b] \times [c, d]$ in the x and y plane. This domain is the

light source of our system from a height chosen depending on what were are trying to map. To reconstruct the surface, we use our intensity data to find $f = -I_z$ and hence solve for P so that we are then able to compute, $\phi(x, y, z) = z + P(x, y)$. Knowing our phase function helps us find the direction of our reflected rays:

$$\nabla\phi(x, y, z) =< P_x, P_y, 1 >$$

From here we first know that the rays hitting the edges of the sensor match the rays hitting the edges of our domain we choose to shoot the rays from. Note that the direction in this case is negative direction of the reflected rays. We want to find a value of t such that (see figure (a)):

$$< a, b, c > -t\nabla\phi =< x_0, y_0, z_0 >$$

Where the point $< a, b, c >$ are the points on sensor where we know the values of all the points and the point x_0 and y_0 are points that match the initial x and y component from the light ray source shot doward to the surface we wish to map. The only unknown point here is z_0. Thus note that:

$$c - t = z_0$$

Now we have the point on the true surface as:

$$f(x_0, y_0) = z_0$$

Focusing on the corner points, we calculate the optical path lengths and choose the minimum between them and call it K. With the K, we will use it to calculate the interior height of the surface. Looking at figure (b), we know K=h_1+h_2+h_3. Since the direction of the reflected ray is $\nabla\phi$ and the direction of the light rays shooting down is $w =< 0, 0, -1 >$, the angle between them can be calculated by

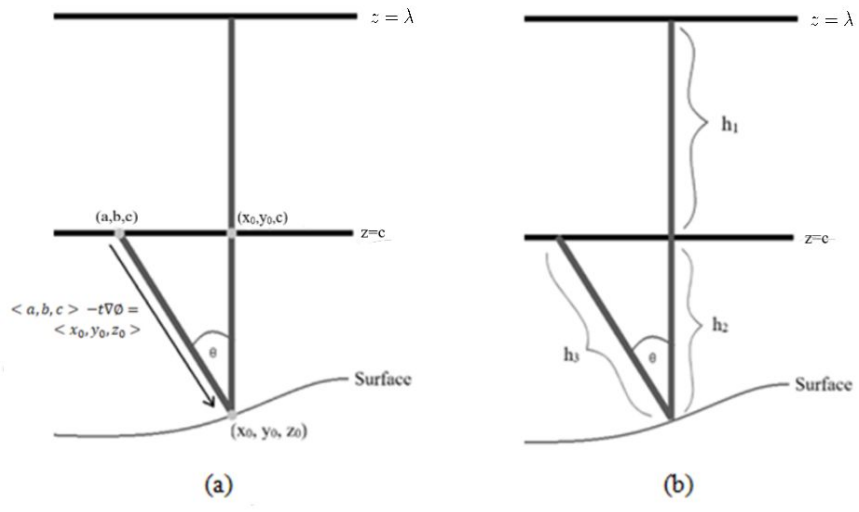
$$\cos\theta = \frac{\nabla\phi \cdot w}{\|\nabla\phi\| \, \|w\|} \Rightarrow \theta = \arccos(\frac{\nabla\phi \cdot w}{\|\nabla\phi\| \, \|w\|})$$

To calculate h_1, h_2, h_3 we use the following

$$h_1 = \lambda - c \qquad h_2 = h_3 \cos\theta \qquad h_3 = \frac{K - h_1}{\cos\theta + 1}$$

So, to get the z value of the surface

$$z_0 = c - h_2$$



(a) Illustration of how to calculate z_0 (b) Illustration of how to calculate $K=h_1+h_2+h_3$

[Zsurf] = phase2surf(p, z_light, z_sensor, x1, x2, y1, y2, h, A1, A2, B1, B2)

The phase2surf function takes the p(x,y) values of the phase function for a given domain and gives the approximation of the surface as Zsurf. The input p represents the $N \times M$ matrix of the $P(x, y)$ values in the domain $[x1, x2] \times [y1, y2]$ of the light source. The height of the light source and the sensor is inputted by z_light and z_sensor, respectively. To create the mesh grid domain of the light source, the function use the input h which represents the mesh size. Once the function get the domain of the light source is made, the function makes the mesh for the domain of the reflected light rays $[A1, A2] \times [B1, B2]$

The function takes the gradient of the p matrix to get the direction of the reflected light rays. For each boundary node, the function calculates the t value using the equations $a - tP_x = x_0$ and $b - tP_y = y_0$. Take the minimum between the two t's as the exact t to solve for z_0 on the boundary.

After finding the height of the surface for the boundary, for each corner of the domain at the sensor height the function calculates the optical path length from the light source to the sensor. This function assume that the minimum optical path length between the corners will be the optical path length of the entire domain, call it K. To find the height of the domain, the function will calculate the values for $K = h_1 + h_2 + h_3$. Then to get the height of the surface the function subtracts z_sensor by h_2.

It is important to note that with this function:

1. The domain of the light rays from the light source and the domain of the light rays hitting the sensor are both rectangular.
2. The parallel light rays shooting down onto the surface have direction < 0, 0, -1>.
3. The gradient of p does not equal to zero or very small to avoid dividing by a zero.

4. The approximated surface is an estimate of the shape of the surface and may have a vertical shift compared to the real surface.

Examples & Figures of Surfaces that Reflects Light with Plane Wavefronts to Form:

Spherical

To capture the spherical object with wavefront:

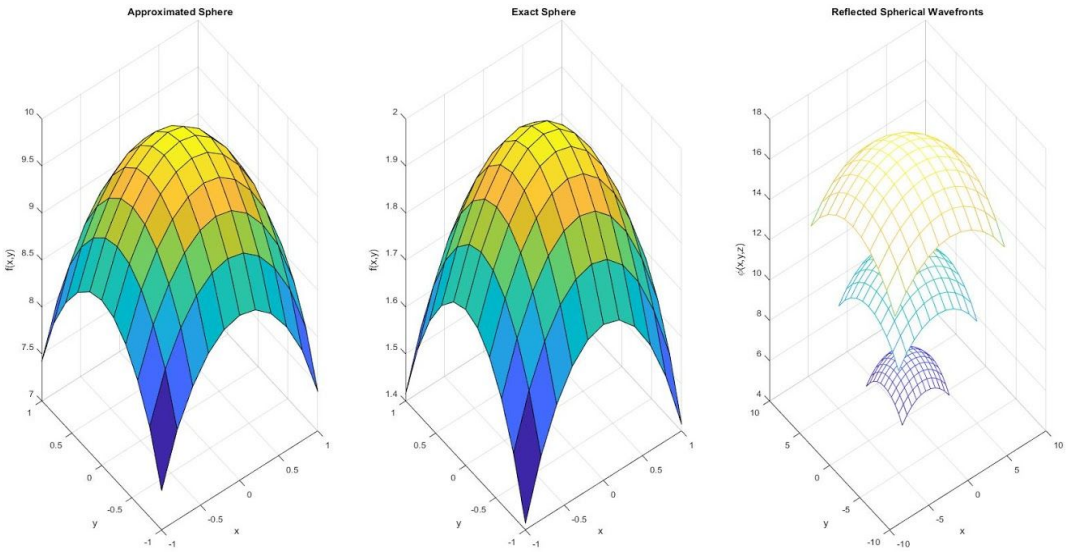
$$\phi(x, y, z)$$

Recall that the gradient of ϕ is:

$$\nabla\phi =< P_x, P_y, 1 >$$

Where $z = \sqrt{k - (x)^2 - (y)^2}$.

Here, the object is centered at the origin. We place our sensor over our wavefront where $k = 3$ at height of $z = 5$. We first map our known surface from a given domain and reflected upwards as to create a wavefront. From here we get our $P(x, y) = k - z$ where our (x, y, z) are known for a certain wavefront we choose. After getting gradients of $P(x, y)$ we follow our gradient and incident vector as to form a triangle to find the sides of the triangle and find the position of z that we are looking for on $f(x, y) = z$. The resulting image are as follows:



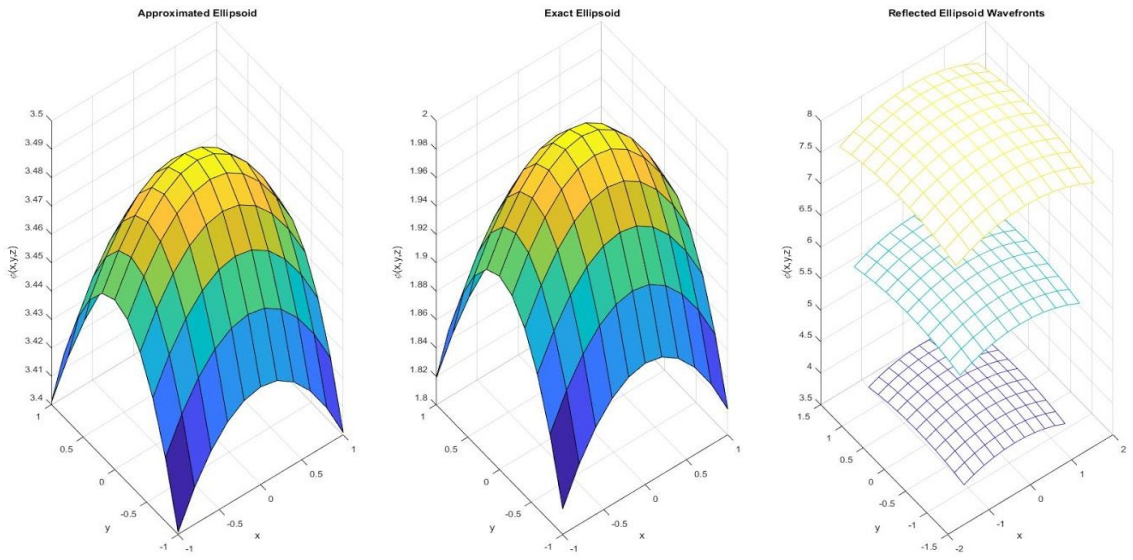
(from left to right) Approximated, Exact and reflected surfaces.

Ellipsoidal

For ellipsoidals, we perform the same method as our spherical example with

$$z = \sqrt{4(1 - \frac{x^2}{9} - \frac{y^2}{16})}$$

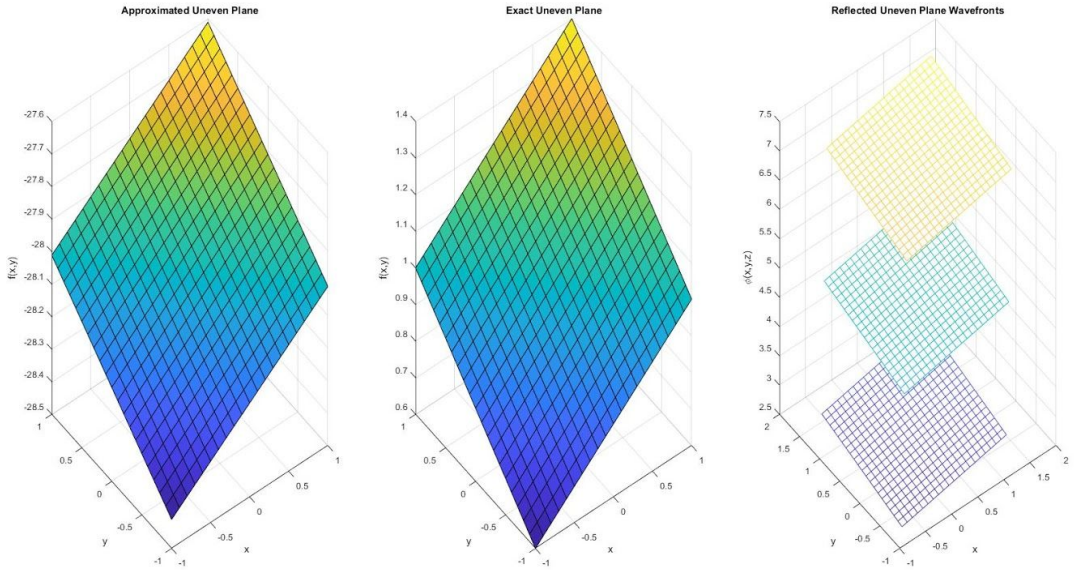
to get images as:



(from left to right) Approximated, Exact and reflected surfaces.

Uneven reflected wavefront

For an uneven surface, $z = 0.2x + 0.2y + 1$



(from left to right) Approximated, Exact and reflected surfaces.

Error

Let the error of the method be given by

$$error = |mean(Approximated - Exact)|$$

Where

$$Exact = Z_{exact} - mean(Z_{exact})$$
$$Approximated = Z_{approximated} - mean(Z_{approximated})$$

For the sphere, the calculated error for the surfaces above are

Spherical : $error = 0.3890$

Ellipsoidal: $error = 0.0196$

Uneven Plane: $error = 2.8319 \times 10^{-15}$

Notice that the error gets smaller as the surface gets flatter and that these surfaces are approximated with exact p values, not approximated with our TIENeumann function.

Limitations and Issues

With this setup, we have the light sensor placed in the path of the incoming light rays before it hits the surface and assume that it does not cause any complications with collecting the intensity of the reflected light rays. Another reason for having the sensor in the path of the downward light rays is that we are assuming that the light rays are paraxial, meaning that the angle between the incident ray and the optical axis (z-axis in this experiment) are small. To ensure that the angle is paraxial, the unknown surface in the experiment should be relatively flat. If the surface is more angled, the experimenter can adjust the mount holding the surface so the reflection will not have a large tilt. If the surface causes the reflecting rays to intersect, the light sensor will have to be placed at a height before the rays cross. If the reflected light rays cross each other rapidly, the experimenter can use a beam expander to stretch the beam to before they cross.

The material of the surface at is being reconstructed will be relatively flat and made of mirrors since the reflection angle of light rays on flat mirrors can be predicted with the Law of Reflection. By the Law of Reflection, the angle between the normal of the surface and the incident light ray, θ_i, is equal to the angle between the normal and the reflected light ray, θ_r (Hecht).

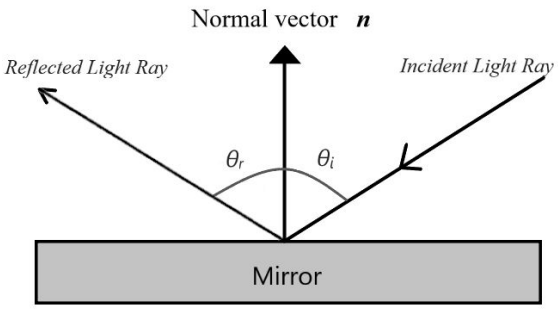


Illustration of the Law of Reflection on mirrors

References

M. Born and E. Wolf, *Principles of Optics*. Pergamon Press 1959.

E. Hecht, *Optics*. Pearson Education Limited 2017.