Investigation of Wave Propagation on Simple 2D Membranes

Trevor Ford
Department of Physics and Astronomy, West Virginia University, Morgantown, WV, 26505, USA

(Dated: 9 December 2020)

GitHub Project Address: https://github.com/TrevorFord01/Physics-301-Project

Abstract: Mechanical waves on simple two-dimensional membranes were investigated via a conglomeration of analytical and numerical techniques. Using the undamped wave equation as a base-level model, the behavior of the time evolution of waves on rectangular and circular membranes was elucidated by a separation of variables technique commonly used in low-level studies involving linear homogeneous partial differential equations. Homogeneous Dirichlet conditions for the membranes' boundaries were used to impose the condition of a "fixed-frame" membrane, such that only interior points are permitted to deviate. Analytical solutions are two-dimensional Fourier (rectangular case) and two-dimensional Fourier-Bessel (circular case) series with coefficients determined by double integration over the domain of the membrane, though the analytical solution for the circular case depended on quantities that can only be determined numerically – namely, specific roots of Bessel functions of the first kind. Using several Python modules (NumPy, SciPy, Matplotlib, etc.), these analytical solutions were rendered as 3D animations to aid in the visualization of their complex forms. A graphical user interface (GUI) was implemented to boost the utility of the project to a wider audience and allows users to set parameters such as membrane shape, size, initial height and velocity distributions, et cetera. The development constructed in this study generates .mp4 and .gif files of solution animations and can be played on any compatible device.

Introduction

A main goal of physics is to determine how systems evolve over time. A two-dimensional membrane is one such system for which analytical and numerical tools have been developed from physical intuitions. The time evolution of a wave, something that is familiar conceptually to practically all people, is described by complicated mathematical functions that often lead to confusion when the problem is introduced to first-time learners. The goals of this investigation were two-fold: 1) determine the forms of analytical solutions that describe wave behavior on rectangular and circular membranes and 2) generate animations of the membranes evolving in time given a set of specified initial conditions.

The analytical solutions were determined by the separation of variables technique, a fundamental method utilized for finding solutions to linear homogeneous partial differential equations (PDEs). This procedure is often one of the first discussed in an undergraduate course in partial differential equations and has many utilities beyond the study of wave motion. This technique returned expressions that were utilized in the animation development. Although the analytical solutions are exact in nature, the circular boundary solutions consist of quantities that could only be determined numerically. This dependence on numerical computations lends the problem well to computational resources.

Several well-known Python modules were implemented in the creation of animations, as well as in the development of a graphical user interface (GUI) that increased utility and expanded the scope of the project to a wider audience. Many errors were addressed in the writing of the animation generator scripts, such as errors that arose due to the mishandling of memory allocation, issues in the display and framerates of animations, et cetera. Several tests were performed on the program to determine its efficiency in calculating necessary series coefficients, which is the bulk of the work that leads to different wave shapes and their evolutions. The development passed all such tests, though that does not exclude the need for further testing to identify and address lurking errors.

This report first addresses the analytical and numerical methods employed for the development of the package also addresses the necessities involved in the construction of the GUI. Upon conclusion discussing the techniques and skills employed for the project, a review and interpretation of the results is performed. These results include still-shots from the figures prepared by the program and address the concerns that arise when representing infinite series with a finite number of terms (Gibbs phenomena). Finally, the culmination of the project is contained within the conclusion and perspectives section. Future developments and present-day applications are also investigated in this section.

Methods

We start our discussion of methods with the introduction of the undamped wave equation[1], a second-order linear and homogeneous PDE. This equation will serve as the basis for all our developments and will be central to our discussion of wave behavior. Let *u* represent the height of the membrane at any given point and let *c* be the speed with which a wave moves on the membrane, then the wave equation constrains the function *u* as

$$\frac{\partial^2 u}{\partial t^2} = c^2 \nabla^2 u \tag{1}$$

The wave equation relates the second time derivative of the height of the membrane to the Laplacian[2] of the height distribution. The Laplacian operator in a general domain space is an operator formed by taking the sum of second partial derivative operators along a set of mutually orthogonal and independent directions. In a two-dimensional domain space, the simplest Laplacian is that expressed in a type of rectangular coordinates by

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \tag{2}$$

If instead however we use a coordinate system whose basis is not independent, such as polar coordinates, then the Laplacian takes a more convoluted form:

$$\nabla^2 = \frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial}{\partial r}\right) + \frac{1}{r^2}\frac{\partial^2}{\partial \theta^2} \tag{3}$$

The Laplacian can be extended to higher dimensions as well, though our goals are accomplished via the use of the Laplacians from equations (2) and (3). For a membrane, which is inherently a two-dimensional object, it is necessary for our Laplacian to take a two-dimensional form. Using these forms of the Laplacian, the wave equation can be written as

$$\frac{\partial^2 u}{\partial t^2} = c^2 \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \tag{4}$$

$$\frac{\partial^2 u}{\partial t^2} = c^2\left(\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial u}{\partial r}\right) + \frac{1}{r^2}\frac{\partial^2 u}{\partial \theta^2}\right) \tag{5}$$

where $u = u(x, y, t)$ in equation (4) and $u = u(r, \theta, t)$ in equation (5).

The wave equation alone is not enough to specify a unique physical problem; boundary conditions must also be imposed to further constrain the solution and describe how it behaves at the membrane's edges. There are two basic types of boundary conditions that can be applied in the case of a two-dimensional membrane, Neumann and Dirichlet conditions[3]. A Neumann condition is one in which reflections of a wave about a boundary do not invert the wave about its amplitude; a Dirichlet condition is one in which reflections of a wave about a boundary result in such an inversion. For the purposes of this investigation, we assumed Dirichlet conditions for the boundary conditions at the membrane edges, which physically corresponds to fixing the ends of the membrane to be a certain height for all time. Considering a rectangular membrane of length L and height H with its bottom left corner fixed at the origin, then the boundary conditions are

$$\begin{gathered} u(0, y, t) = 0 \\ u(L, y, t) = 0 \\ u(x, 0, t) = 0 \\ u(x, H, t) = 0 \end{gathered} \tag{6}$$

Since these boundary conditions constrain the function to be equal to zero at specified points, they are considered homogeneous boundary conditions. The boundary conditions for the circular membrane take a more unique form due to periodicity, continuity, and singularity concerns[4]. Assuming a circular membrane with radius r_0, the boundary conditions are:

$$\begin{gathered} |u(0, \theta, t)| < \infty \\ u(r_0, \theta, t) = 0 \\ u(r, -\pi, t) = u(r, \pi, t) \\ \frac{\partial u}{\partial \theta}(r, -\pi, t) = \frac{\partial u}{\partial \theta}(r, \pi, t) \end{gathered} \tag{7}$$

Now that the necessary equations to specify the problems have been determined, we focus our attention on methods to solve the problem. We will be following the standard separation of variables method but will be brushing over the finer details and calculations, which are outlined in many external resources[5,6]. The basis for this method is the assumption that the solution follows as a product form, for example assuming $u(r, \theta, t) = R(r)\Theta(\theta)T(t)$ or $u(x, y, t) = X(x)Y(y)T(t)$. Upon making these substitutions, the rectangular problem simplifies into three ordinary differential equations and two boundary value problems:

$$\begin{cases} \dfrac{d^2 T}{dt^2} = -c^2\lambda_1 T \\ \dfrac{d^2 X}{dx^2} = -\lambda_2 X \\ \dfrac{d^2 Y}{dy^2} = (\lambda_2 - \lambda_1)Y \end{cases} \qquad \begin{gathered} X(0) = 0 \\ X(L) = 0 \\ Y(0) = 0 \\ Y(H) = 0 \end{gathered} \tag{8}$$

The circular problem also takes a similar collapsed form:

$$\begin{cases} \dfrac{d^2 T}{dt^2} = -c^2\lambda_1 T \\ r^2\dfrac{d^2 R}{dr^2} + r\dfrac{dR}{dr} = (\lambda_2 - \lambda_1 r^2) \\ \dfrac{d^2 \Theta}{d\theta^2} = -\lambda_2 \Theta \end{cases} \qquad \begin{gathered} |R(0)| < \infty \\ R(r_0) = 0 \\ \Theta(-\pi) = \Theta(\pi) \\ \frac{d\Theta}{d\theta}(-\pi) = \frac{d\Theta}{d\theta}(\pi) \end{gathered} \tag{9}$$

where the constants λ_1 and λ_2 are known as separation constants and will be interpreted in the boundary value problems as eigenvalues. Note that the radial ordinary differential equation in (9) is a variant of Bessel's equation[7], and as such will have Bessel function[8] solutions. The separation of variables procedure combined with the principle of superposition for linear homogeneous PDEs reveal that the solution to (8) is

$$u(x,y,t)=\sum_{n=1}^{\infty}\sum_{m=1}^{\infty}\left(A_{m,n}\cos\left(\sqrt{\frac{m^2}{H^2}+\frac{n^2}{L^2}}\pi ct\right)+B_{m,n}\sin\left(\sqrt{\frac{m^2}{H^2}+\frac{n^2}{L^2}}\pi ct\right)\right)\sin\left(\frac{n\pi x}{L}\right)\sin\left(\frac{m\pi y}{H}\right) \qquad (10)$$

The solution to (9) is:

$$\begin{aligned}u(x,y,t)=\sum_{n=0}^{\infty}\sum_{m=1}^{\infty}\Bigg(&A_{m,n}\cos(n\theta)\cos\left(\frac{cz_{n,m}t}{r_0}\right)+B_{m,n}\cos(n\theta)\sin\left(\frac{cz_{n,m}t}{r_0}\right)+C_{m,n}\sin(n\theta)\cos\left(\frac{cz_{n,m}t}{r_0}\right)\\&+D_{m,n}\sin(n\theta)\sin\left(\frac{cz_{n,m}t}{r_0}\right)\Bigg)J_n\left(\frac{z_{n,m}r}{r_0}\right)\end{aligned} \qquad (11)$$

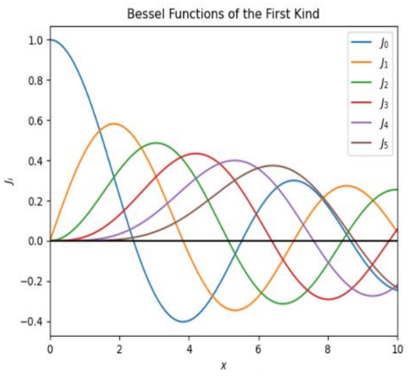


Figure 1, Bessel Functions of the First Kind – plot of the first five Bessel functions that arise when solving the circular membrane problem.

where J_n is the nth Bessel function of the first kind and $z_{n,m}$ is the mth root of the nth Bessel function of the first kind. These solutions are two-dimensional Fourier (10) and two-dimensional Fourier-Bessel (11) series[9]. The roots of the Bessel functions in equation (11) must be determined numerically, so expressing the solution via computational methods is the most reasonable approach.

To determine the coefficients for these infinite series, we apply initial conditions that describe the height and velocity distributions at the time $t=0$. Let $f(x,y)=u(x,y,0)$ and $g(x,y)=\frac{\partial u}{\partial t}(x,y,0)$, then the coefficients in (10) are given by:

$$A_{m,n}=\frac{4}{HL}\int_0^H\int_0^L f(x,y)\sin\left(\frac{n\pi x}{L}\right)\sin\left(\frac{m\pi y}{H}\right)dx\,dy \qquad (12)$$

$$B_{m,n}=\frac{4}{\pi c\sqrt{m^2L^2+n^2H^2}}\int_0^H\int_0^L g(x,y)\sin\left(\frac{n\pi x}{L}\right)\sin\left(\frac{m\pi y}{H}\right)dx\,dy \qquad (13)$$

Further, if we let $f(r,\theta)=u(r,\theta,0)$ and $g(r,\theta)=\frac{\partial u}{\partial t}(r,\theta,0)$, then the coefficients in (11) are given by:

$$A_{m,0}=\frac{1}{\pi r_0^2\left(J_1(z_{0,m})\right)^2}\int_0^{r_0}\int_0^{2\pi} rf(r,\theta)J_0\left(\frac{z_{0,m}r}{r_0}\right)d\theta\,dr \qquad (14)$$

$$B_{m,0}=\frac{1}{\pi r_0cz_{0,m}\left(J_1(z_{0,m})\right)^2}\int_0^{r_0}\int_0^{2\pi} rg(r,\theta)J_0\left(\frac{z_{0,m}r}{r_0}\right)d\theta\,dr \qquad (15)$$

$$A_{m,n}=\frac{2}{\pi r_0^2\left(J_{n+1}(z_{n,m})\right)^2}\int_0^{r_0}\int_0^{2\pi} rf(r,\theta)\cos(n\theta)J_n\left(\frac{z_{n,m}r}{r_0}\right)d\theta\,dr \qquad (16)$$

$$B_{m,n} = \frac{2}{\pi r_0 c z_{n,m} \left(J_{n+1}(z_{n,m})\right)^2} \int_0^{r_0} \int_0^{2\pi} r g(r,\theta) \cos(n\theta) J_n\left(\frac{z_{n,m} r}{r_0}\right) d\theta\, dr \tag{17}$$

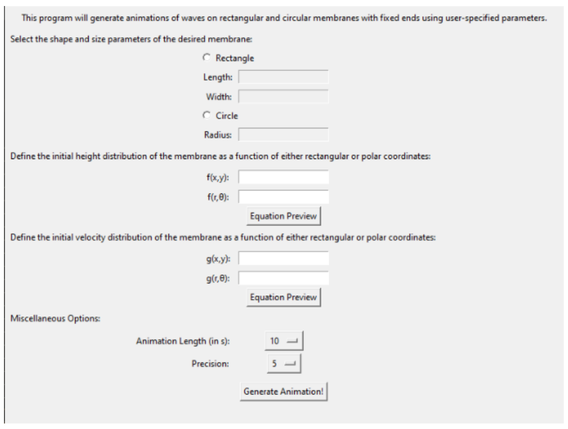
$$C_{m,n} = \frac{2}{\pi r_0^2 \left(J_{n+1}(z_{n,m})\right)^2} \int_0^{r_0} \int_0^{2\pi} r f(r,\theta) \sin(n\theta) J_n\left(\frac{z_{n,m} r}{r_0}\right) d\theta\, dr \tag{18}$$

$$D_{m,n} = \frac{2}{\pi r_0 c z_{n,m} \left(J_{n+1}(z_{n,m})\right)^2} \int_0^{r_0} \int_0^{2\pi} r g(r,\theta) \sin(n\theta) J_n\left(\frac{z_{n,m} r}{r_0}\right) d\theta\, dr \tag{19}$$

Also note that the constants $C_{m,0}$ and $D_{m,0}$ for the Fourier-Bessel series will both be zero because the sine function in the integrand becomes zero when $n = 0$. Using these results given here, we move on to discuss the connection of the theory with computational methods required to generate solution animations.

This development is implemented with Python making use of the NumPy, SciPy, SymPy, iPywidgets, Matplotlib, TkInter, MoviePy, and Pillow modules. The first scripts to implement are the animation generations. We use the `linspace()` and `meshgrid()` functions available in NumPy to generate the domain of the membranes. We set the precision (i.e. the number of Fourier terms along a given direction), the wave speed, and create functions that represent the initial height and velocity distributions of the membrane. To determine the Fourier coefficients, we make use of the Gaussian quadrature[10] method over a 2D space with the `dblquad()` function available in SciPy. These Fourier coefficients are stored in arrays to be called upon later when implementing the time evolution.

To implement the functional forms of equations (10) and (11), we break the computation up into a few smaller chunks – each functional piece of the solution is represented by a different chunk. We compute the results of applying the domain through the chunks and then multiply the results together in order to preserve some computational power at the expense of some speed. We then use the animation module of Matplotlib to generate animations with a set number of frames and certain time delay for frame updates. This returns a Matplotlib animation object that can be saved as a .gif file by the Pillow module.


This program will generate animations of waves on rectangular and circular membranes with fixed ends using user-specified parameters.

Select the shape and size parameters of the desired membrane:

( ) Rectangle

Length:

Width:

( ) Circle

Radius:

Define the initial height distribution of the membrane as a function of either rectangular or polar coordinates:

f(x,y):

f(r,θ):

Equation Preview

Define the initial velocity distribution of the membrane as a function of either rectangular or polar coordinates:

g(x,y):

g(r,θ):

Equation Preview

Miscellaneous Options:

Animation Length (in s): 10

Precision: 5

Generate Animation!

Figure 2, GUI – image of graphical user interface developed for the program.

The final development was the construction of the graphical user interface (GUI) to extend the utility and appeal of the program to a wider audience. The main implementation was developed by use of the TkInter module, with specific user-input data collected from text fields, radiobutton selections, and buttons. The "Equation Preview" selection present on the GUI, shown in Figure 2, takes inspiration from the integral-calculator.com equation preview field, though requires the use of buttons rather than real-time updates to a whitespace. The precision input relates the number of Fourier series and Fourier-Bessel series coefficients calculated and utilized by the program. The animation length setting in the GUI allows users to choose how long of an animation they wish to generate. Upon activating the

"Generate Animation!" button, the program collects all the user-input data and passes it to the animation generation files. The animation objects are returned to the GUI script and then saves the animation as both .gif and .mp4 files in the same folder.

Results and Discussion

The results from the program are difficult to include in a report, since they are inherently time-evolving figures. However, it is possible to show time traces for certain specific sets of conditions. We will demonstrate the utility of the program on circular, rectangular, and square membranes by their initial time plots, since if these plots are appropriate it is highly probable that the resulting wave evolution is also appropriate.

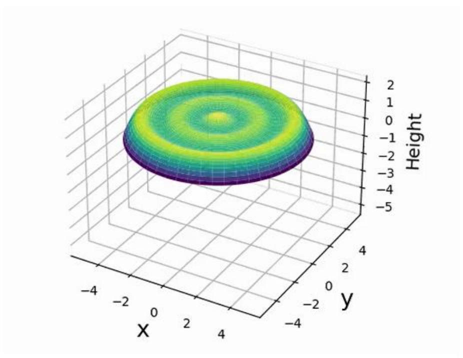


Figure 3, Circular Membrane – initial distribution plot of a waveform on a circular membrane.

We start by demonstrating the initial trace for a circular membrane of radius 5 with initial height distribution $f(r,\theta)=1$ and initial velocity distribution $g(r,\theta)=1$. Note the presence of the Gibbs phenomenon near the edges of the membrane – this is because we have approximated an infinite series by a finite number of terms. There are also ripple-like structures within the membrane that are also present for the same reason. However, it is clear to see that the initial distribution would converge to 1 over the whole membrane as we increase the number of terms we use in the numerical calculation.

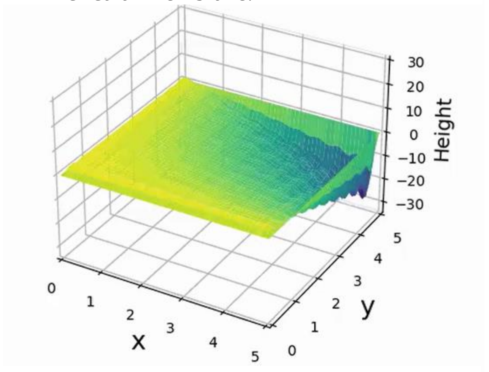


Figure 4, Square Membrane – initial distribution plot of a waveform on a square membrane.

The next membrane to investigate will be the square case – we consider a square with side length 5, initial height distribution $f(x,y)=3-xy$, and initial velocity distribution $g(x,y)=0$. Again, we see Gibbs phenomena at the boundary edges that gets more severe the greater the distance between the membrane initial height function and the membrane base becomes. However, it is also still clear in this case that the initial distribution would be satisfied if we were able to take an infinite number of terms for the Fourier series. The precision input used for this generation was set to 20 in order to maximize the output and demonstrate the sharpness of the Gibbs phenomenon.

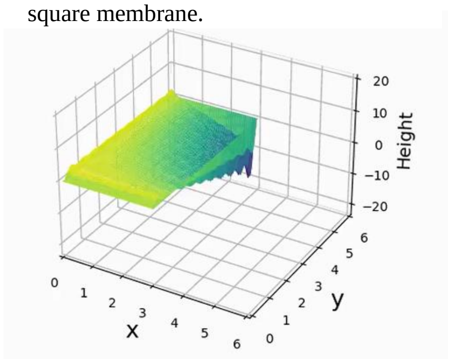


Figure 5, Rectangular Membrane – initial distribution plot of a waveform on a rectangular membrane.

Finally, we investigate the case of a rectangle with different side lengths. If we let the length be 3 and the width be 6 and apply the same conditions as were used for the above square membrane, then we again see that the distribution functions match the output plot. The Gibbs phenomenon in this case is also just about as prevalent as in the square case, despite the function taking on much less-extreme values. The results here demonstrate the accuracy in the generation of the Fourier and Fourier-Bessel coefficients, confirming the theory behind the waveforms and the numerical implementation. We have also confirmed that the appropriate shapes and sizes of membranes were generated when using input from the GUI. It was not demonstrated here, but the GUI can also handle instance of exponential and trigonometric function inputs without the prefacing of the functions with a specified module call. This is made possible due to an additional NumPy star import command in the animation generation codes.

It is important to note that the GUI takes functional inputs but cannot recognize characters such as $3x$ to mean $3 * x$ – it is essential that users use the appropriate Python basic mathematical operations when giving functional inputs. Many programs that allow functional inputs are known to use the carat symbol, ^, for exponentiation, but for this program the ** operator must be used. The equation input utility of this program is phenomenal and allows for many more initial conditions than can be achieved with the basic addition, subtraction, multiplication, division, etc.

The main implementation of the animation generation scripts had many issues involving memory allocation, hence why list comprehensions are present in places where NumPy operations would naturally be more efficient to carry out. As a result, the code lost a bit of speed, but did not sacrifice any precision thankfully. The results seem to match with what is expected, and the videos generated by the program seem to be remarkably accurate.

Conclusions and Perspectives

Our goal of developing a program to assist people in the visualization of wave equation solutions on 2D membranes was a success. Not only were we able to generate the animations and implement a rudimentary GUI, but we were also able to obtain reasonably accurate results and allow for different parameters to describe a wave's motion on these membranes. The work here has many applications to any kind of system that experiences time-evolution according to the undamped wave equation, and not just those in two dimensions. For example, the time evolution of light from a classical optics standpoint[11] is also governed by the same wave equation used in this project. Any field that utilizes wave behavior or uses this canonical PDE (the wave equation) will find utility in the results of this development.

Future improvements to the project are ample. In addition to basic procedures such as code restructuring to regain some of the lost speed, we can also make improvements to the GUI itself. For instance, as it is currently the GUI cannot handle piecewise function inputs, despite such inputs being relatively important in a general study of partial differential equations. The program files themselves are able to handle such piecewise inputs by using conditional statements inside the definitions of the initial distribution functions, but it is much harder for a user to follow this procedure than it is to simply type the desired equations into a text box.

In this project, we also only focused on the Dirichlet boundary conditions, but it is completely possible to investigate Neumann boundaries and the even more general Robin boundaries. Such implementations would give users even more capability to study the evolution of waves. Further, we also only focused on the method of separation of variables for this project, since it is the least mathematically and numerically complex, but there are numerous other methods that can be employed that have much more application than the separation of variables procedures does. For example, the separation of variables method works well for finding solutions to the wave equation on rectangular and circular membranes but does not work well in practically any other case. This poses a challenge, since there are certainly more shapes that a membrane can have. Other methods based on finite difference, finite volumes, etc. would be able to handle these more exotic boundary shapes such as triangles, pentagons, hexagons, etc. and would bring even more clarity on the topic of wave evolution to the program operator.

Overall, the goals we set to achieve were accomplished and even more computational experience was gained by the completion of this investigation. There are many possible exciting future developments and modern applications of this project, each of which are not performed here due to time restriction and lack of more general knowledge on the subject matter. It is with great hope that we can return to this study and gradually make improvements to its utility, efficiency, and appeal. The necessity for visualization tools is ever-present as more people in today's world learn upper-level mathematics, and this project has served to try to bridge the gap between the abstract concepts of partial differential equations and modern learners.

Bibliography

[1]Nave C. R. "The Wave Equation" [Online]. 1999. http://hyperphysics.phy-astr.gsu.edu/hbase/Waves/waveq.html [Oct. 2020].

[2]"Laplace Operator" [Online]. 2020. https://en.wikipedia.org/wiki/Laplace_operator [Oct. 2020].

[3]Hattori H. *Partial Differential Equations: Methods, Applications, and Theories*, pp. 33. Singapore: World Scientific Publishing Co. Pte. Ltd., 2013.

[4]"Dirichlet and Heat Problems in Polar Coordinates" [Online]. 2020. http://www.math.ttu.edu/~gilliam/ttu/s10/m3351_s10/c14_2d_disk_heat [Dec. 2020]

[5]Haberman R. *Applied Partial Differential Equations with Fourier Series and Boundary Value Problems*, 5e, pp. 137-140,272-282,295-307. Boston, MA, USA: Pearson Education, Inc., 2019.

[6]"Vibrations of a Circular Membrane" [Online]. 2020. https://en.wikipedia.org/wiki/Vibrations_of_a_circular_membrane [Nov. 2020].

[7]Svirin, A. "Bessel Differential Equation" [Online]. 2020. https://www.math24.net/bessel-differential-equation/ [Nov. 2020].

[8]"Bessel Functions" [Online]. 2020. https://en.wikipedia.org/wiki/Bessel_function [Dec. 2020].

[9]"Fourier-Bessel Series" [Online]. 2020. https://encyclopediaofmath.org/wiki/Fourier-Bessel_series [Dec. 2020].

[10]Landau, R. H., Bordeianu, C. C., & Páez, M. J. *A Survey of Computational Physics: Introductory Computational Science*, pp. 97-102. Princeton, NJ: Princeton Univ. Press., 2008.

[11]Hecht, E. *Optics*. Harlow: Pearson., 2017.