Rose-Hulman Undergraduate Mathematics Journal

Volume 16
Issue 1

Article 9

Identifying a Coefficient Matrix for Numerical Approximations to the Wave Equation in Two Dimensions

Morgan M. Brown
University of Mary Washington

Rose-Hulman Undergraduate Mathematics Journal

Identifying a Coefficient Matrix for Numerical Approximations to the Wave Equation in Two Dimensions

Morgan M. Brown[a]

Volume 16, No. 1, Spring 2015

Sponsored by

Rose-Hulman Institute of Technology
Department of Mathematics
Terre Haute, IN 47803
Email: mathjournal@rose-hulman.edu
http://www.rose-hulman.edu/mathjournal

[a]University of Mary Washington

Rose-Hulman Undergraduate Mathematics Journal
Volume 16, No. 1, Spring 2015

Identifying a Coefficient Matrix for Numerical Approximations to the Wave Equation in Two Dimensions

Morgan M. Brown

Abstract. In this paper we develop a coefficient matrix to be used in numerical approximation methods which model the wave equation in two dimensions. In particular, we will briefly introduce the centered difference approximation method. Next, we explain the derivation of the system of equations into which the problem is transformed in order to utilize such a method. Then, we introduce a set of rules to generate the generalized coefficient matrix for use in approximating the wave equation for any number of unknowns per axis. Finally, we write MATLAB code which uses our matrix to solve said approximations, and then use it in a real world application. The desired outcome of this project has been achieved: to generalize the coefficient matrix used in the system of equations which approximates the wave equation in two dimensions so that its algorithm may be used in MATLAB code for any number of unknowns.

Acknowledgements: The author would like to thank Dr. Jangwoon Lee for his guidance.

1 Introduction

One of the primary goals of mathematics is to model physical phenomena in order to gain a better understanding of the world in which we live. This goal is often realized by examining a number of equations which describe changes over time. One such equation, known as the wave equation, models the motion pattern of a number of objects and events, in as high as three dimensions and beyond. This equation holds great importance in subjects ranging from music theory to seismology. Herein, we investigate the ways in which the wave equation can be solved using numerical approximation methods, which offer a highly accurate solution in much less time and with much less computation than can be offered by an exact solution.

In this paper, we develop a coefficient matrix to be used in the two-dimensional wave equation. In Section 2, we introduce the wave equation and its initial and boundary conditions. Section 3 discusses the method of solving the equation with numerical approximations, and Section 4 introduces the algorithm for calculating the coefficient matrix used in these approximations. Section 5 describes the MATLAB program used to compute solutions to the equation and in Section 6 we offer some example output.

The main focus of this manuscript is to introduce the aforementioned algorithm, which is previously unenumerated in the literature. A generalized form of the coefficient matrix for numerical approximations to the one-dimenisonal wave equation has often been presented, as it has a relatively straightforward and simple construction. Additionally, explicit examples for the two-dimensional case exist in many textbooks. However, this is the first occurence of an algorithm which allows for the coefficient matrix to be built given any number of unknowns. The algorithm provides mathematicians with a means to create this matrix using a method other than brute force.

2 Definition

By examining the motion of a vibrating plane, we can derive the two-dimensional wave equation

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

where c^2 is a constant and $\nabla^2 u$ is the Laplacian, with initial conditions

$$u(x, y, 0) = \alpha(x, y), \ \frac{\partial u}{\partial t}(x, y, 0) = \beta(x, y)$$

and homogeneous boundary conditions

$$u(0, y, t) = u(L, y, t) = u(x, 0, t) = u(x, H, t) = 0,$$

where L is the horizontal length of the rectangular membrane and H is the vertical length (Haberman 2004 [2]).

3 Numerical Approximations

Often times in practice, it is not possible to obtain the exact solution to a problem. Thus, we will use a numerical approximation for this equation. Consider the centered difference approximation to $\frac{d^2f}{dx^2}(x_0)$ (Burden 2005 [1]):

$$\frac{d^2f}{dx^2}(x_0) \approx \frac{f(x_0+\Delta x)+f(x_0-\Delta x)-2f(x_0)}{(\Delta x)^2}.$$

We will be using $u(x,y,t)$ instead of $f(x)$. Hence, we first will introduce a new notation:

$$u^m_{(j,k)} \equiv u(x_j, y_k, t_m)$$

with

$$\begin{aligned} x_j + \Delta x &= x_{j+1} \\ y_k + \Delta y &= y_{k+1} \\ t_m + \Delta t &= t_{m+1}. \end{aligned}$$

That is, $x_j = j\Delta x$, $y_k = k\Delta y$, and $t_m = m\Delta t$ where j is the number of steps in the x direction, k is the number of steps in the y direction, and m is the number of time steps. Note that $j = 1, 2, ..., n-1$, $n = \frac{L}{\Delta x}$ such that n is an integer, $k = 1, 2, ..., p-1$, $p = \frac{H}{\Delta y}$ such that p is an integer, and $m = 1, 2, ...$

By applying this to both sides of the equation, we may obtain the finite difference approximation to the two-dimensional wave equation:

$$u^{m+1}_{(j,\,k)} - 2u^m_{(j,\,k)} + u^{m-1}_{(j,\,k)} = s\left(u^m_{(j+1,\,k)} + u^m_{(j-1,\,k)} + u^m_{(j,\,k+1)} + u^m_{(j,\,k-1)} - 4u^m_{(j,\,k)}\right)$$

where $s = \left(\frac{c\Delta t}{\Delta x}\right)^2$.

3.1 Systems of Equations

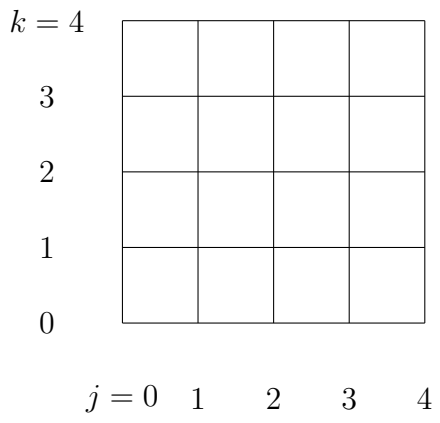
We want to be able to recursively define each step, so that we are finding the solution at a given time using the object's position at previous times. We can now begin to write a numerical approximation. Solving recursively for time $m+1$ gives

$$\begin{aligned} u^{m+1}_{(j,\,k)} &= s\left(u^m_{(j+1,\,k)} + u^m_{(j-1,\,k)} + u^m_{(j,\,k+1)} + u^m_{(j,\,k-1)} - 4u^m_{(j,\,k)}\right) + 2u^m_{(j,\,k)} - u^{m-1}_{(j,\,k)} \\ &= su^m_{(j+1,\,k)} + su^m_{(j-1,\,k)} + su^m_{(j,\,k+1)} + su^m_{(j,\,k-1)} - 4su^m_{(j,\,k)} + 2u^m_{(j,\,k)} - u^{m-1}_{(j,\,k)} \\ &= su^m_{(j+1,\,k)} + su^m_{(j-1,\,k)} + su^m_{(j,\,k+1)} + su^m_{(j,\,k-1)} + (2-4s)u^m_{(j,\,k)} - u^{m-1}_{(j,\,k)}. \end{aligned}$$

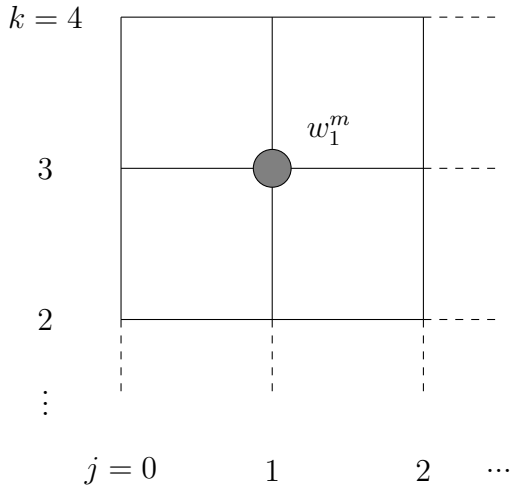
In order to write this as a system of equations that can be translated into matrix form, we will introduce a new method of indexing. As opposed to indexing the space variations as

$(j,\ k)$, we will index with a value $i = j + (n-1-k)(n-1)$, where n is one more than the number of unknowns per axis (Strange 2011 [3]). So, our new notation will be $u^m_{(j,\ k)} \equiv w^m_i$. Notice that this could give rise to several issues. None of the indices can equal zero or less, nor can they equal the maximum number of indices, $(n-1)\cdot(n-1)$, or higher. If, for instance, we have nine total unknowns, calculating an i value of ten would go out of bounds of our numbering system. In these cases, we will still use the function u, meaning the function w is not always applicable. Using this information and our boundary conditions, we are able to write a system of equations in matrix form to help solve for the membrane's vertical position at given locations along the mesh and at certain times.

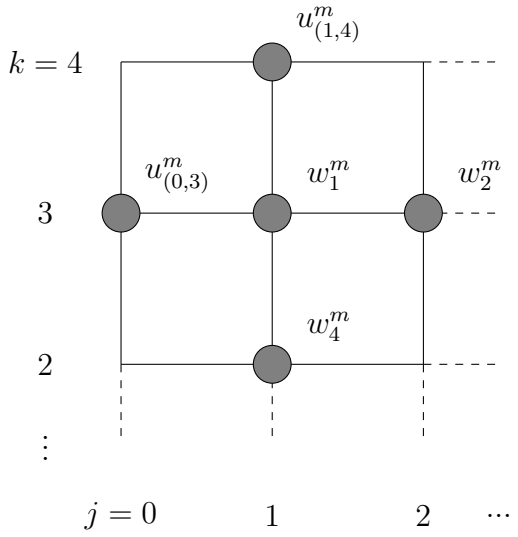
To visualize the indexing system, consider the following. Let $n = 4$. Then, there will be $(n-1)\cdot(n-1) = 3\cdot 3 = 9$ indexed locations, with i ranging from 1 to 9. We have:



Clearly, $k = 0, 4$ and $j = 0, 4$ are the boundaries of the object, and will be equal to zero as per our boundary conditions. Note the location for the point $u^{m+1}_{(1,\ 3)} \equiv w^m_1$.



The value here is constructed with the values from the following points at time $m\Delta t$:



Take note of the pattern that occurs. We took data from the point itself, the points directly above and below, as well as the points directly to the left and right. Note that in general form, we have

$$\begin{pmatrix} w_1^{m+1} \\ w_2^{m+1} \\ \vdots \\ w_{(n-1)\cdot(n-1)}^{m+1} \end{pmatrix} = A \begin{pmatrix} w_1^{m} \\ w_2^{m} \\ \vdots \\ w_{(n-1)\cdot(n-1)}^{m} \end{pmatrix} - \begin{pmatrix} w_1^{m-1} \\ w_2^{m-1} \\ \vdots \\ w_{(n-1)\cdot(n-1)}^{m-1} \end{pmatrix}$$

with coefficient matrix A varying depending on number of unknowns per axis, n.

4 The General Coefficient Matrix

A pattern for the coefficient matrix is becoming apparent. In our quest to write a generalized coefficient matrix A, we can make several observations:

1. The value $2-4s$ is present in every location along the main diagonal.

2. The value s is present along the diagonal directly below the main diagonal, except in the $n^{\text{th}}, n+(n-1)^{\text{th}}, n+2(n-1)^{\text{th}}$... rows, where there is a 0. This is because these entries correspond to locations on the mesh that fall on a boundary (we have the zero boundary condition).

3. The value s is present along the diagonal directly above the main diagonal, except in the $(n-1)^{\text{th}}, 2(n-1)^{\text{th}}, 3(n-1)^{\text{th}}$... rows, where there is a 0. This is because these entries correspond to locations on the mesh that fall on a boundary (we have the zero boundary condition).

4. The value s is present along the diagonal $(n-1)$ below the main diagonal, i.e., starting in the n^{th} row and first column and proceeding in a diagonal fashion to the end of the matrix.

5. The value s is present along the diagonal $(n-1)$ above the main diagonal, i.e., starting in the first row and n^{th} column and proceeding in a diagonal fashion to the end of the matrix.

6. All other entries are zero.

Our matrix A in general form can be written as follows, where the underlined and bolded parts serve only to provide a row and column number:

$$\begin{pmatrix}
 & \underline{\mathbf{1}} & \underline{\mathbf{2}} & \underline{\mathbf{3}} & & \underline{\mathbf{n-2}} & \underline{\mathbf{n-1}} & \underline{\mathbf{n}} & \underline{\mathbf{n+1}} & \underline{\mathbf{n+2}} & \underline{\mathbf{n+3}} & \\
\underline{\mathbf{1}} & 2-4s & s & 0 & ... & 0 & 0 & s & 0 & 0 & 0 & ... \\
\underline{\mathbf{2}} & s & 2-4s & s & ... & 0 & 0 & 0 & s & 0 & 0 & ... \\
\underline{\mathbf{3}} & 0 & s & 2-4s & ... & 0 & 0 & 0 & 0 & s & 0 & ... \\
 & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\
\underline{\mathbf{n-1}} & 0 & 0 & 0 & ... & s & 2-4s & 0 & 0 & 0 & 0 & ... \\
\underline{\mathbf{n}} & s & 0 & 0 & ... & 0 & 0 & 2-4s & s & 0 & 0 & ... \\
\underline{\mathbf{n+1}} & 0 & s & 0 & ... & 0 & 0 & s & 2-4s & s & 0 & ... \\
 & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots &
\end{pmatrix}.$$

This pattern will continue in the same way for the entire matrix, which is of size $(n-1)\cdot(n-1)$ by $(n-1)\cdot(n-1)$. Let the row number be represented by r, and the column number by c. This matrix follows these general rules:

1. If $r = c$ for every $r, c \in \{1, 2, ..., (n-1)\cdot(n-1)\}$, then we have $A[r, c] = 2 - 4s$.

2. If $r = c + 1$ for every $r \in \{2, 3, ..., (n-1)\cdot(n-1)\}, c \in \{1, 2, ..., (n-1)\cdot(n-1)-1\}$, then we have $A[r, c] = s$ **except** under the following circumstance:

   - For all $r = n, n+(n-1) = 2n-1, n+2(n-1) = 3n-2, ...$ $\in \{2, 3, ..., (n-1)\cdot(n-1)\}$ and $c = r-1$, $A[r, c] = 0$.

3. If $r = c - 1$ for every $r \in \{1, 2, ..., (n-1)\cdot(n-1)-1\}, c \in \{2, 3, ..., (n-1)\cdot(n-1)\}$, then we have $A[r, c] = s$ **except** under the following circumstance:

   - For all $r = n-1, 2(n-1) = 2n-2, 3(n-1) = 3n-3, ... \in \{1, 2, ..., (n-1)\cdot(n-1)-1\}$ and $c = r+1$, $A[r, c] = 0$

4. If $r = c + (n-1)$ for every $r \in \{n, n+1, ..., (n-1)\cdot(n-1)\}, c \in \{1, 2, ..., (n-1)\cdot(n-2)\}$, then we have $A[r, c] = s$.

5. If $r = c - (n-1)$ for every $r \in \{1, 2, ..., (n-1)\cdot(n-2)\}, c \in \{n, n+1, ..., (n-1)\cdot(n-1)\}$, then we have $A[r, c] = s$.

6. All other entries are 0.

To check this general form, recall the problem from Section 3.1 when $n = 4$. By brute force, we find the following coefficient matrix:

$$\begin{pmatrix} 2-4s & s & 0 & s & 0 & 0 & 0 & 0 & 0 \\ s & 2-4s & s & 0 & s & 0 & 0 & 0 & 0 \\ 0 & s & 2-4s & 0 & 0 & s & 0 & 0 & 0 \\ s & 0 & 0 & 2-4s & s & 0 & s & 0 & 0 \\ 0 & s & 0 & s & 2-4s & s & 0 & s & 0 \\ 0 & 0 & s & 0 & s & 2-4s & 0 & 0 & s \\ 0 & 0 & 0 & s & 0 & 0 & 2-4s & s & 0 \\ 0 & 0 & 0 & 0 & s & 0 & s & 2-4s & s \\ 0 & 0 & 0 & 0 & 0 & s & 0 & s & 2-4s \end{pmatrix}.$$

Now, we create a matrix by implementing our guidelines. Note that since $n = 4$, we have $(n-1)\cdot(n-1) = 3 \cdot 3 = 9$.

1. If $r = c$ for every $r, c \in \{1, 2, ..., 9\}$, then we have $A[r, c] = 2 - 4s$. So, we have $A[1, 1] = A[2, 2] = A[3, 3] = A[4, 4] = A[5, 5] = A[6, 6] = A[7, 7] = A[8, 8] = A[9, 9] = 2 - 4s$.

$$\begin{pmatrix} 2-4s & ? & ? & ? & ? & ? & ? & ? & ? \\ ? & 2-4s & ? & ? & ? & ? & ? & ? & ? \\ ? & ? & 2-4s & ? & ? & ? & ? & ? & ? \\ ? & ? & ? & 2-4s & ? & ? & ? & ? & ? \\ ? & ? & ? & ? & 2-4s & ? & ? & ? & ? \\ ? & ? & ? & ? & ? & 2-4s & ? & ? & ? \\ ? & ? & ? & ? & ? & ? & 2-4s & ? & ? \\ ? & ? & ? & ? & ? & ? & ? & 2-4s & ? \\ ? & ? & ? & ? & ? & ? & ? & ? & 2-4s \end{pmatrix}$$

2. If $r = c + 1$ for every $r \in \{2, 3, ..., 9\}, c \in \{1, 2, ..., 8\}$, then we have $A[r, c] = s$ **except** under the following circumstance:

- For $r = \{4, 7\}$ and $c = r - 1 = \{4 - 1, 7 - 1\} = \{3, 6\}$, $A[r, c] = 0$.

Thus, we have $A[2, 1] = A[3, 2] = A[5, 4] = A[6, 5] = A[8, 7] = A[9, 8] = s$ and $A = [4, 3] = A[7, 6] = 0$.

$$\begin{pmatrix} 2-4s & ? & ? & ? & ? & ? & ? & ? & ? \\ s & 2-4s & ? & ? & ? & ? & ? & ? & ? \\ ? & s & 2-4s & ? & ? & ? & ? & ? & ? \\ ? & ? & 0 & 2-4s & ? & ? & ? & ? & ? \\ ? & ? & ? & s & 2-4s & ? & ? & ? & ? \\ ? & ? & ? & ? & s & 2-4s & ? & ? & ? \\ ? & ? & ? & ? & ? & 0 & 2-4s & ? & ? \\ ? & ? & ? & ? & ? & ? & s & 2-4s & ? \\ ? & ? & ? & ? & ? & ? & ? & s & 2-4s \end{pmatrix}$$

3. If $r = c - 1$ for every $r \in \{1, 2, ..., 8\}, c \in \{2, 3, ..., 9\}$, then we have $A[r, c] = s$ **except** under the following circumstance:

   - For $r = \{3, 6\}$ and $c = r + 1 = \{3 + 1, 6 + 1\} = \{4, 7\}$, $A[r, c] = 0$.

   So, we have $A[1, 2] = A[2, 3] = A[4, 5] = A[5, 6] = A[7, 8] = A[8, 9] = s$ and $A[3, 4] = A[6, 7] = 0$.

$$\begin{pmatrix} 2-4s & s & ? & ? & ? & ? & ? & ? & ? \\ s & 2-4s & s & ? & ? & ? & ? & ? & ? \\ ? & s & 2-4s & 0 & ? & ? & ? & ? & ? \\ ? & ? & 0 & 2-4s & s & ? & ? & ? & ? \\ ? & ? & ? & s & 2-4s & s & ? & ? & ? \\ ? & ? & ? & ? & s & 2-4s & 0 & ? & ? \\ ? & ? & ? & ? & ? & 0 & 2-4s & s & ? \\ ? & ? & ? & ? & ? & ? & s & 2-4s & s \\ ? & ? & ? & ? & ? & ? & ? & s & 2-4s \end{pmatrix}$$

4. If $r = c + 3$ for every $r \in \{4, 5, ..., 9\}, c \in \{1, 2, ..., 6\}$, then we have $A[r, c] = s$. In other words, $A[4, 1] = A[5, 2] = A[6, 3] = A[7, 4] = A[8, 5] = A[9, 6] = s$.

$$\begin{pmatrix} 2-4s & s & ? & ? & ? & ? & ? & ? & ? \\ s & 2-4s & s & ? & ? & ? & ? & ? & ? \\ ? & s & 2-4s & 0 & ? & ? & ? & ? & ? \\ s & ? & 0 & 2-4s & s & ? & ? & ? & ? \\ ? & s & ? & s & 2-4s & s & ? & ? & ? \\ ? & ? & s & ? & s & 2-4s & 0 & ? & ? \\ ? & ? & ? & s & ? & 0 & 2-4s & s & ? \\ ? & ? & ? & ? & s & ? & s & 2-4s & s \\ ? & ? & ? & ? & ? & s & ? & s & 2-4s \end{pmatrix}$$

5. If $r = c - 3$ for every $r \in \{1, 2, ..., 6\}, c \in \{4, 5, ..., 9\}$, then we have $A[r, c] = s$. Then, we have $A[1, 4] = A[2, 5] = A[3, 6] = A[4, 7] = A[5, 8] = A[6, 9] = s$.

$$\begin{pmatrix} 2-4s & s & ? & s & ? & ? & ? & ? & ? \\ s & 2-4s & s & ? & s & ? & ? & ? & ? \\ ? & s & 2-4s & 0 & ? & s & ? & ? & ? \\ s & ? & 0 & 2-4s & s & ? & s & ? & ? \\ ? & s & ? & s & 2-4s & s & ? & s & ? \\ ? & ? & s & ? & s & 2-4s & 0 & ? & s \\ ? & ? & ? & s & ? & 0 & 2-4s & s & ? \\ ? & ? & ? & ? & s & ? & s & 2-4s & s \\ ? & ? & ? & ? & ? & s & ? & s & 2-4s \end{pmatrix}$$

6. All other entries are 0.

$$\begin{pmatrix} 2-4s & s & 0 & s & 0 & 0 & 0 & 0 & 0 \\ s & 2-4s & s & 0 & s & 0 & 0 & 0 & 0 \\ 0 & s & 2-4s & 0 & 0 & s & 0 & 0 & 0 \\ s & 0 & 0 & 2-4s & s & 0 & s & 0 & 0 \\ 0 & s & 0 & s & 2-4s & s & 0 & s & 0 \\ 0 & 0 & s & 0 & s & 2-4s & 0 & 0 & s \\ 0 & 0 & 0 & s & 0 & 0 & 2-4s & s & 0 \\ 0 & 0 & 0 & 0 & s & 0 & s & 2-4s & s \\ 0 & 0 & 0 & 0 & 0 & s & 0 & s & 2-4s \end{pmatrix}$$

Again, this is identical to the matrix we derived using our equations. It is reasonable now to conclude that our algorithm for the general form is indeed valid and will work to determine the coefficient matrix for any n.

5 Computer Simulations

Our goal is to find solutions to this system. So, we want to write a program in MATLAB which can solve the equation on a larger scale, when calculations become too difficult to perform by hand. To do this, we are going to solve $w^{m+1} = Aw^m - w^{m-1}$ where w^{m+1}, w^m, and w^{m-1} are vectors, and with A equal to the general coefficient matrix as defined in the previous section.

To solve the system of equations for any time step $m + 1$, we need to be able to initialize the procedure. That is, we must have matrices representing w_i^0 and w_i^1 so that we may find w_i^2, and then w_i^3, and so on. But clearly, the equation we outlined does not work for finding the zeroth and first time steps, as that would require information about negative time steps, which do not exist. So, we want to determine a general way to write these initial systems. Recall our initial conditions:

$$u(x, y, 0) = \alpha(x, y), \ \frac{\partial u}{\partial t}(x, y, 0) = \beta(x, y)$$

Now, it is clear that we will obtain w_i^0 simply by using $\alpha(x_j, y_k)$. That is, we break $i = j + (n-1-k)(n-1)$ into its j and k values, which represent the number of times we have incremented Δx and Δy, respectively. So, we construct the matrix at time zero by using $w_i^0 = \alpha(j\Delta x, k\Delta y)$, where $i = j + (n-1-k)(n-1)$.

Now, we are faced with the challenge of constructing the matrix at the first time step. Recall the wave equation:

$$\frac{\partial^2 u}{\partial t^2} = c^2 \left(\frac{\partial^2 u}{\partial x^2}(x_j, y_k, t_m) + \frac{\partial^2 u}{\partial y^2}(x_j, y_k, t_m) \right).$$

And, consider this at time zero:

$$\frac{\partial^2 u}{\partial t^2} = c^2 \left(\frac{\partial^2 u}{\partial x^2}(x_j, y_k, 0) + \frac{\partial^2 u}{\partial y^2}(x_j, y_k, 0) \right).$$

Also recall that $u(x, y, 0) = \alpha(x, y)$, and assume there exist α_{xx} and α_{yy}. Then, the previous is equal to

$$c^2 \left(\frac{\partial^2 \alpha}{\partial x^2}(x_j, y_k) + \frac{\partial^2 \alpha}{\partial y^2}(x_j, y_k) \right) = c^2(\alpha_{xx}(x_j, y_k) + \alpha_{yy}(x_j, y_k)).$$

We know that the Taylor series expansion of a function is as follows:

$$f(x+h) = f(x) + hf\,'(x) + \frac{h^2}{2} f\,''(x) + \frac{h^3}{6} f\,'''(\mu), \;\; x < \mu < x + h.$$

This can easily be applied to our equation in the time dimension. We want to determine an equation for the first time step, $t_1 = t_0 + \triangle t = \triangle t$. In the form of $x + h$, this means $x = 0$ and $h = \triangle t$. So:

$$u(x_j, y_k, t_1) = \alpha(x_j, y_k) + \Delta t \frac{\partial u}{\partial t}(x_j, y_k, 0) + \frac{\Delta t^2}{2} \frac{\partial^2 u}{\partial t^2}(x_j, y_k, 0) + \frac{\Delta t^3}{6} \frac{\partial^3 u}{\partial t^3}(x_j, y_k, \mu).$$

Given $\dfrac{\partial u}{\partial t}(x, y, 0) = \beta(x, y)$ and the way we wrote the wave equation in terms of the function α, we can write

$$u(x_j, y_k, t_1) = \alpha(x_j, y_k) + \Delta t \beta(x_j, y_k) + \frac{\Delta t^2}{2} c^2(\alpha_{xx}(x_j, y_k) + \alpha_{yy}(x_j, y_k)) + \frac{\triangle t^3}{6} \frac{\partial^3 u}{\partial t^3}(x_j, y_k, \mu).$$

We can approximate this by saying

$$u(x_j, y_k, t_1) \approx \alpha(x_j, y_k) + \triangle t \beta(x_j, y_k) + \frac{\triangle t^2}{2} c^2(\alpha_{xx}(x_j, y_k) + \alpha_{yy}(x_j, y_k)).$$

Thus, we have developed a method for determining the plane's position at the first time step, using its position at the zeroth time step as well as the initial condition β, and the

second partial derivatives of α. Hence, given any wave equation problem, we can use the system of matrices to solve for any time step.

In order to solve problems like this both efficiently and quickly, we will develop a program using MATLAB. This program will take in the following parameters: $\alpha_{xx}, \alpha_{yy}, \beta, c, \Delta t, \Delta x$ (which will be equal to Δy by assumption), the maximum number of time steps, and the length of each side of the plane. Then, the output of the program will be a large matrix. Each row will correspond to an i value for the function w, which can then be decomposed into the (x, y) location on the plane. Each column will correspond to a time step, with the first column being time 0, the second column being Δt, and so on.

As an example run of the program, consider the wave equation acting on a plane that is 1.5 units long by 1.5 units tall. Let $\Delta x = \Delta y = 0.5$. Then, we will have three sub-planes along each axis, meaning $n = 3$, with a total of $(n-1)\cdot(n-1) = 2 \cdot 2 = 4$ mesh points within the plane that need values. Let us calculate up to $2\Delta t$ with a time step size of 0.1 units. Let c be 1. Recall that we have $s = \left(\frac{c\Delta t}{\Delta x}\right)^2$. Then in this case, $s = \left(\frac{1(0.1)}{(0.5)}\right)^2 = 0.04$. Let $\alpha(x, y) = x^2 + y^2$ and $\beta(x, y) = x + y$. The output was as follows:

```
>> W2d( 3, 0.1, 0.5, 0.5, 2, 1, 'alpha1', 'alphaXX1', 'al

s =

    0.0400


A =

    1.8400    0.0400    0.0400         0
    0.0400    1.8400         0    0.0400
    0.0400         0    1.8400    0.0400
         0    0.0400    0.0400    1.8400


ans =

    1.2500    1.4200    1.4764
    2.0000    2.2200    2.1984
    0.5000    0.6200    0.7544
    1.2500    1.4200    1.4764
```

6 Implementation

Now that we know we have an accurate program, we can use it to solve implementations of the wave equation which are more difficult to solve by hand. That is, we can calculate for a greater number of subplanes n and a greater number of time steps, as well as under more complicated initial conditions, and so on.

Recall that the goal of the two-dimensional wave equation is to model the vibrations of a plane. One of the best examples of a plane whose behavior obeys the wave equation is a simple trampoline. Consider a square trampoline which is one unit long on each side. When at rest, the trampoline is perfectly flat. So, its initial displacement at time zero may be modeled by

$$\alpha(x, y) = 0.$$

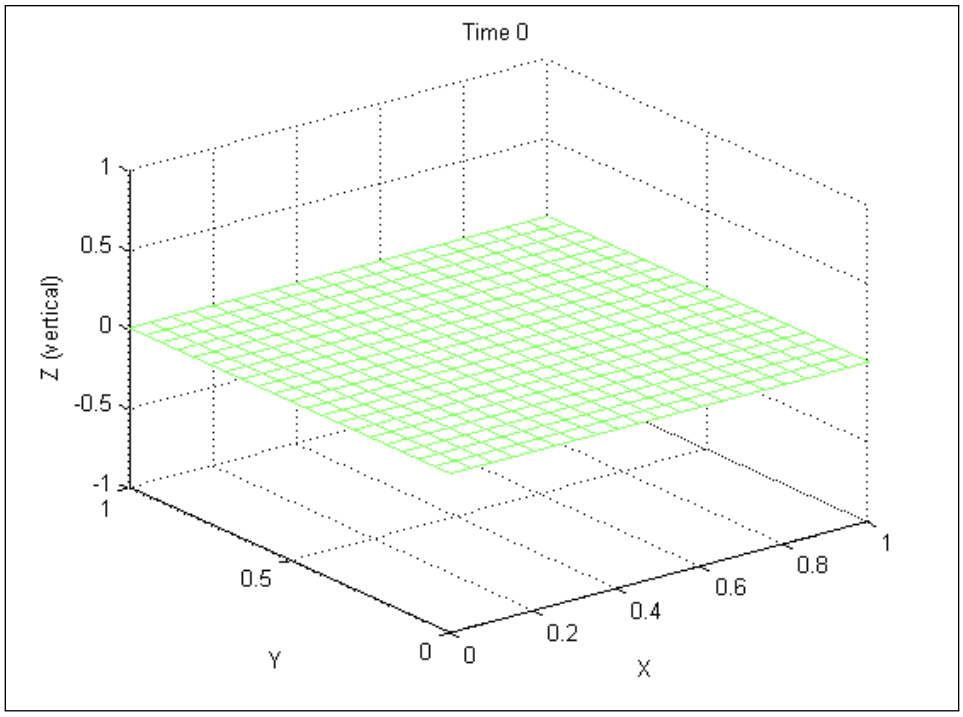
Now, imagine that something is dropped into the center of the trampoline, setting the plane in motion and causing its velocity to be modeled by

$$\beta(x, y) = 4\sin(x) + 4\sin(y).$$

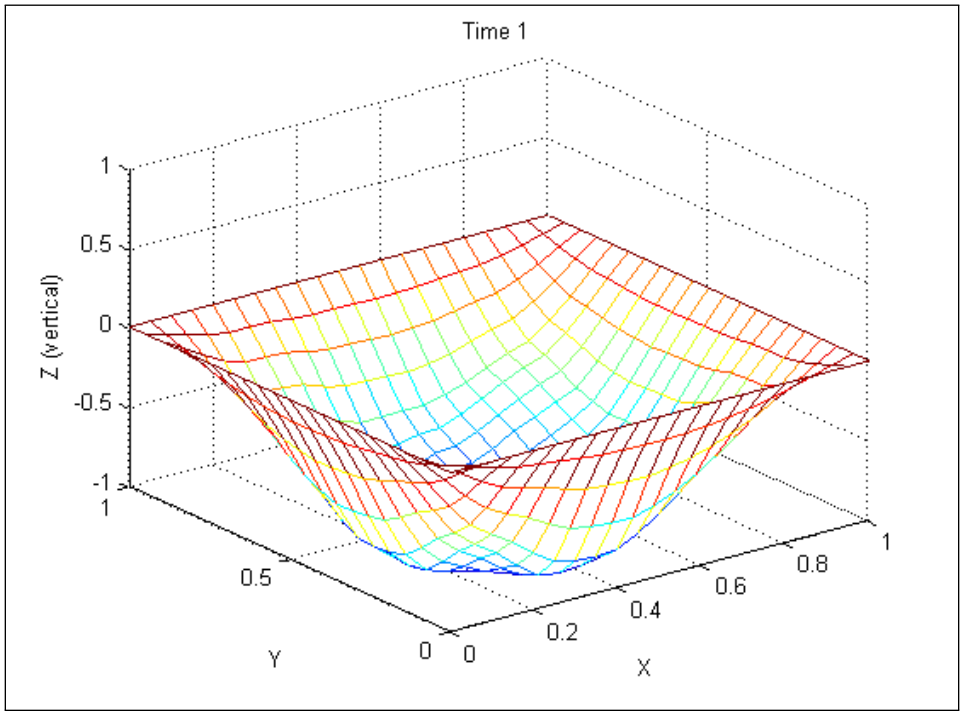
We want to create a thorough mesh with lots of data points, so we will let $n = 20$ and we will let c be 1. Then, Δx and Δy will be 0.05 each. We will let Δt be 0.01. This gives an s value of $\left(\frac{c\Delta t}{\Delta x}\right)^2 = \left(\frac{0.01}{0.05}\right)^2 = (0.2)^2 = 0.04$.

Clearly, this problem would be both challenging and tedious to solve by hand, especially for a significant amount of time. But, using the MATLAB program, we can find the trampoline's position very quickly. On a large scale, having MATLAB return only a table of values is impractical. So, we developed a program which generates a visual representation of the trampoline. The program takes in a single column from the original MATLAB program as one of its parameters. Then, it rearranges the data entries into a form representative of the trampoline by dissecting the row number into an (x, y) location by using our formula for i. Then, because the location $(0, 0)$ on our grid is in the bottom left (See Section 3.1 for clarification) while MATLAB considers $(0, 0)$ to be the top left when graphing, the program also had to invert the rows. It is easy to check this program's graphing accuracy against pen and paper calculations by deconstructing i values to find where each vertical value should be on the graph.

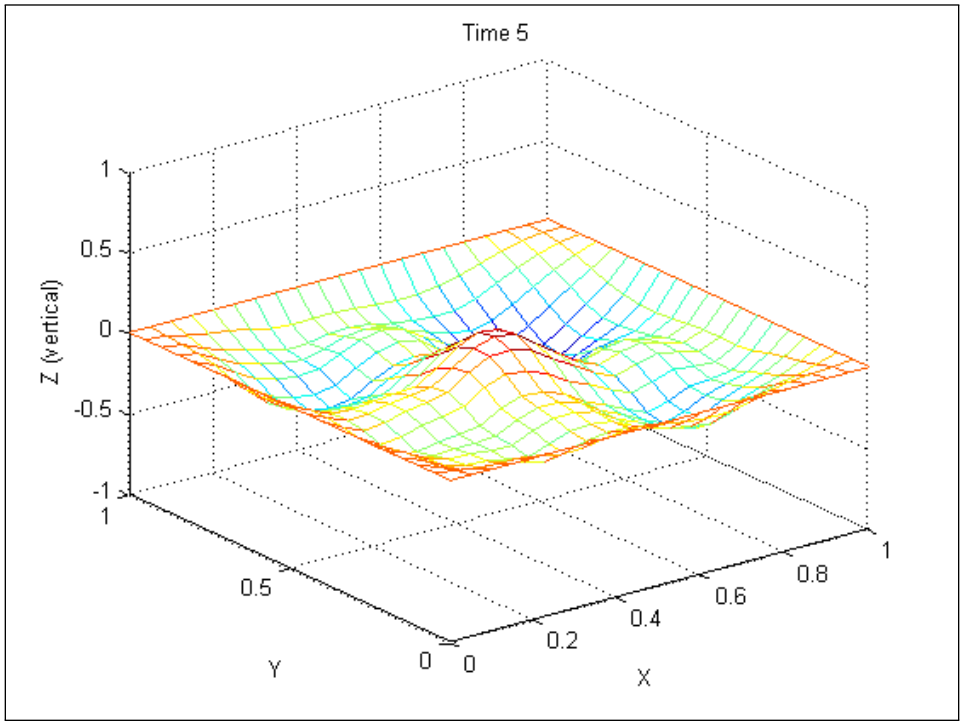
In the following images, the x and y axes represent horizontal locations along the trampoline, and the z axis represents the trampoline's vertical displacement. At time zero, the trampoline is perfectly flat due to the initial displacement.



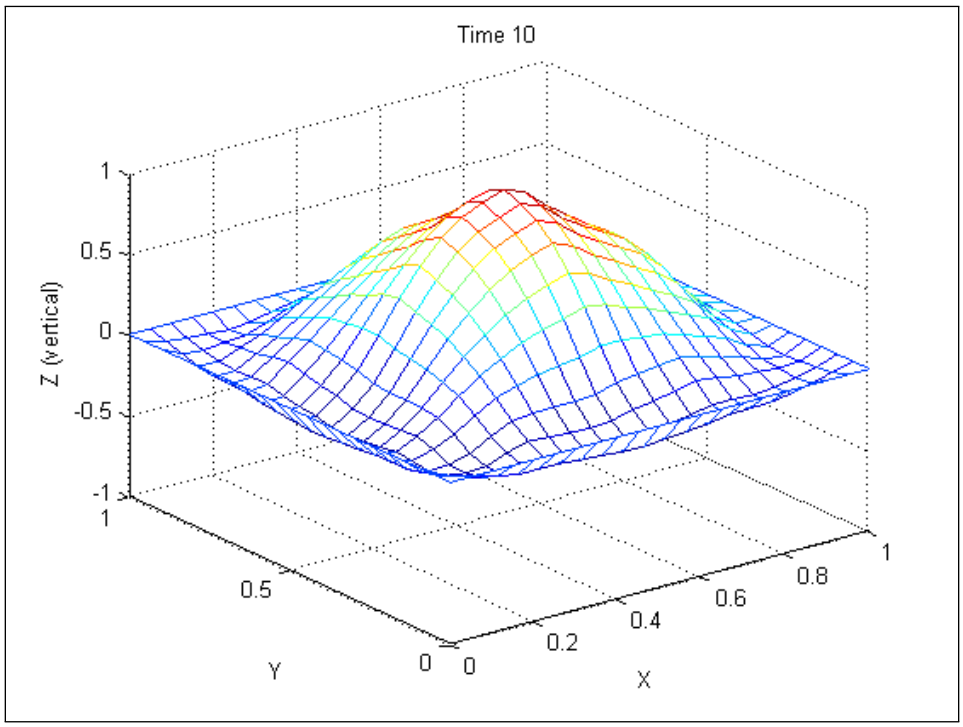
Since we are not using MATLAB only to do difficult calculations quickly, but also to do large numbers of iterations of these calculations in short amounts of time, consider the trampoline's surface after a large number of time steps. That is, we may be interested in the plane's position after one full second, or $100\Delta t$,



its position after five seconds, or $500\Delta t$,



or even its position after ten seconds, or $1,000\Delta t$.



References

[1] Burden, Richard L., and J. Douglas Faires. Numerical Analysis. 8th ed. N.p.: Thomson Brooks/Cole, 2005. Print.

[2] Haberman, Richard. Applied Partial Differential Equations: With Fourier Series and Boundary Value Problems. 4th ed. Upper Saddle River, NJ: Pearson Prentice Hall, 2004. Print.

[3] Strange, Erin L. "Computational Models of the Diffusion Equation." Thesis. University of Mary Washington, 2011. Print.